Álgebra II Práctica (clase 18)

Iván Sadofschi Costa

Universidad de Buenos Aires

26 de Junio de 2020

Prerrequisitos

Para leer estas diapositivas se recomienda haber leído el apunte teórico hasta 3.11.

$\mathbb{C}[\mathbb{Z}_n]$

Sea g un generador de $\mathbb{Z}_n$.

$\mathbb{C}[\mathbb{Z}_n]$

Sea g un generador de $\mathbb{Z}_n$.

Los elementos de $\mathbb{C}[\mathbb{Z}_n]$ tienen la forma $\sum_{i=0}^{n-1} a_i g^i$.

$\mathbb{C}[\mathbb{Z}_n]$

Sea g un generador de $\mathbb{Z}_n$.

Los elementos de $\mathbb{C}[\mathbb{Z}_n]$ tienen la forma $\sum_{i=0}^{n-1} a_i g^i$.

Sea $\xi = e^{2\pi i/n}$.

$\mathbb{C}[\mathbb{Z}_n]$

Sea g un generador de $\mathbb{Z}_n$.

Los elementos de $\mathbb{C}[\mathbb{Z}_n]$ tienen la forma $\sum_{i=0}^{n-1} a_i g^i$.

Sea $\xi = e^{2\pi i/n}$. Podemos considerar para cada $k = 0, \ldots, n-1$ el morfismo de anillos

$$\begin{aligned} \varphi_k : \mathbb{C}[\mathbb{Z}_n] &\to \mathbb{C} \\ \sum_{i=0}^{n-1} a_i g^i &\mapsto \sum_{i=0}^{n-1} a_i \left(\xi^k\right)^i. \end{aligned}$$

$\mathbb{C}[\mathbb{Z}_n]$

Sea g un generador de $\mathbb{Z}_n$.

Los elementos de $\mathbb{C}[\mathbb{Z}_n]$ tienen la forma $\sum_{i=0}^{n-1} a_i g^i$.

Sea $\xi = e^{2\pi i/n}$. Podemos considerar para cada $k = 0, \ldots, n-1$ el morfismo de anillos

$$\begin{aligned} \varphi_k &: \mathbb{C}[\mathbb{Z}_n] \to \mathbb{C} \\ \sum_{i=0}^{n-1} a_i g^i &\mapsto \sum_{i=0}^{n-1} a_i \left(\xi^k\right)^i. \end{aligned}$$

Podemos considerar el producto de estos morfismos

$$\varphi : \mathbb{C}[\mathbb{Z}_n] \to \mathbb{C}^n.$$

$\mathbb{C}[\mathbb{Z}_n]$

Sea g un generador de $\mathbb{Z}_n$.

Los elementos de $\mathbb{C}[\mathbb{Z}_n]$ tienen la forma $\sum_{i=0}^{n-1} a_i g^i$.

Sea $\xi = e^{2\pi i/n}$. Podemos considerar para cada $k = 0, \ldots, n-1$ el morfismo de anillos

$$\begin{aligned} \varphi_k &: \mathbb{C}[\mathbb{Z}_n] \to \mathbb{C} \\ \sum_{i=0}^{n-1} a_i g^i &\mapsto \sum_{i=0}^{n-1} a_i \left(\xi^k\right)^i. \end{aligned}$$

Podemos considerar el producto de estos morfismos

$$\varphi : \mathbb{C}[\mathbb{Z}_n] \to \mathbb{C}^n.$$

Vamos a ver que este morfismo es biyectivo y por lo tanto es un isomorfismo.

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

Basta ver que $\mathcal{F}$ es isomorfismo.

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

Basta ver que $\mathcal{F}$ es isomorfismo.
¿Qué pasa si calculamos $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$?

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

Basta ver que $\mathcal{F}$ es isomorfismo.
¿Qué pasa si calculamos $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$? La ℓ-ésima coordenada de $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$ es

$$\sum_{k=0}^{n-1} \left(\sum_{i=0}^{n-1} a_i \xi^{ki} \right) \left(\xi^\ell\right)^k =$$

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

Basta ver que $\mathcal{F}$ es isomorfismo.
¿Qué pasa si calculamos $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$? La ℓ-ésima coordenada de $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$ es

$$\sum_{k=0}^{n-1} \left(\sum_{i=0}^{n-1} a_i \xi^{ki} \right) \left(\xi^\ell\right)^k = \sum_{i=0}^{n-1} a_i \sum_{k=0}^{n-1} \xi^{(i+\ell)k}$$

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

Basta ver que $\mathcal{F}$ es isomorfismo.
¿Qué pasa si calculamos $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$? La ℓ-ésima coordenada de $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$ es

$$\begin{aligned} \sum_{k=0}^{n-1} \left(\sum_{i=0}^{n-1} a_i \xi^{ki} \right) \left(\xi^\ell\right)^k &= \sum_{i=0}^{n-1} a_i \sum_{k=0}^{n-1} \xi^{(i+\ell)k} \\ &= \begin{cases} n a_{n-\ell} & \text{si } \ell > 0 \\ n a_0 & \text{si } \ell = 0. \end{cases} \end{aligned}$$

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

Basta ver que $\mathcal{F}$ es isomorfismo.
¿Qué pasa si calculamos $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$? La ℓ-ésima coordenada de $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$ es

$$\begin{aligned} \sum_{k=0}^{n-1} \left(\sum_{i=0}^{n-1} a_i \xi^{ki} \right) \left(\xi^\ell\right)^k &= \sum_{i=0}^{n-1} a_i \sum_{k=0}^{n-1} \xi^{(i+\ell)k} \\ &= \begin{cases} n a_{n-\ell} & \text{si } \ell > 0 \\ n a_0 & \text{si } \ell = 0. \end{cases} \end{aligned}$$

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

Basta ver que $\mathcal{F}$ es isomorfismo.
¿Qué pasa si calculamos $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$? La ℓ-ésima coordenada de $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$ es

$$\begin{aligned} \sum_{k=0}^{n-1} \left(\sum_{i=0}^{n-1} a_i \xi^{ki} \right) \left(\xi^\ell\right)^k &= \sum_{i=0}^{n-1} a_i \sum_{k=0}^{n-1} \xi^{(i+\ell)k} \\ &= \begin{cases} n a_{n-\ell} & \text{si } \ell > 0 \\ n a_0 & \text{si } \ell = 0. \end{cases} \end{aligned}$$

Esto permite recuperar los a_i!

φ es isomorfismo

Consideramos la transformación $\mathbb{C}$-lineal $\mathcal{F}\colon \mathbb{C}^n \to \mathbb{C}^n$ dada por

$$(a_0, \ldots, a_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} a_i \left(\xi^0\right)^i, \sum_{i=0}^{n-1} a_i \left(\xi^1\right)^i, \ldots, \sum_{i=0}^{n-1} a_i \left(\xi^{n-1}\right)^i \right).$$

Basta ver que $\mathcal{F}$ es isomorfismo.
¿Qué pasa si calculamos $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$? La ℓ-ésima coordenada de $\mathcal{F}(\mathcal{F}(a_0, \ldots, a_{n-1}))$ es

$$\begin{aligned} \sum_{k=0}^{n-1} \left(\sum_{i=0}^{n-1} a_i \xi^{ki} \right) \left(\xi^\ell\right)^k &= \sum_{i=0}^{n-1} a_i \sum_{k=0}^{n-1} \xi^{(i+\ell)k} \\ &= \begin{cases} n a_{n-\ell} & \text{si } \ell > 0 \\ n a_0 & \text{si } \ell = 0. \end{cases} \end{aligned}$$

Esto permite recuperar los a_i! Luego $\mathcal{F}$ es isomorfismo.

φ es isomorfismo (aclaración)

En la cuenta anterior usamos que

$$\sum_{k=0}^{n-1} \xi^{(i+\ell)k} = \begin{cases} \frac{\xi^{(i+\ell)n}-1}{\xi^{i+\ell}-1} & \text{si } n \nmid i+\ell \\ n & \text{si } n \mid i+\ell. \end{cases}$$

Volviendo a $\mathbb{C}[\mathbb{Z}_n]$

Esto nos permitió escribir a $\mathbb{C}[\mathbb{Z}_n]$ como producto de copias del anillo $\mathbb{C}$.

Volviendo a $\mathbb{C}[\mathbb{Z}_n]$

Esto nos permitió escribir a $\mathbb{C}[\mathbb{Z}_n]$ como producto de copias del anillo $\mathbb{C}$.

Sabíamos que esto se puede hacer (por Maschke y Wedderburn).

Volviendo a $\mathbb{C}[\mathbb{Z}_n]$

Esto nos permitió escribir a $\mathbb{C}[\mathbb{Z}_n]$ como producto de copias del anillo $\mathbb{C}$.

Sabíamos que esto se puede hacer (por Maschke y Wedderburn).

Pero esto nos da un isomorfismo concreto.

Volviendo a $\mathbb{C}[\mathbb{Z}_n]$

Esto nos permitió escribir a $\mathbb{C}[\mathbb{Z}_n]$ como producto de copias del anillo $\mathbb{C}$.

Sabíamos que esto se puede hacer (por Maschke y Wedderburn).

Pero esto nos da un isomorfismo concreto. Esta construcción se parece un poco a la demostración de Maschke, no?

Transformada discreta de Fourier

La función $\mathcal{F}$ se llama la Transformada Discreta de Fourier (DFT) y tiene un montón de aplicaciones!

Transformada discreta de Fourier

La función $\mathcal{F}$ se llama la Transformada Discreta de Fourier (DFT) y tiene un montón de aplicaciones!

Vimos que la inversa de $\mathcal{F}$ (casi) es $\mathcal{F}$.

Transformada discreta de Fourier

La función $\mathcal{F}$ se llama la Transformada Discreta de Fourier (DFT) y tiene un montón de aplicaciones!

Vimos que la inversa de $\mathcal{F}$ (casi) es $\mathcal{F}$.

Entonces esta operación no solo es reversible sino que calcular la inversa no es más dificil que aplicar $\mathcal{F}$.

Aplicación: multiplicar polinomios

Una de las aplicaciones clásicas es multiplicar polinomios rápidamente.

Aplicación: multiplicar polinomios

Una de las aplicaciones clásicas es multiplicar polinomios rápidamente.

Sean $f = \sum_{i=0}^{m} a_i x^i$, $g = \sum_{i=0}^{m} b_i x^i$ dos polinomios de grado m.

Aplicación: multiplicar polinomios

Una de las aplicaciones clásicas es multiplicar polinomios rápidamente.

Sean $f = \sum_{i=0}^{m} a_i x^i$, $g = \sum_{i=0}^{m} b_i x^i$ dos polinomios de grado m.

Para multiplicar f y g hay que distribuir:

$$f \cdot g = \sum_{i=0}^{m} a_i x^i \cdot \sum_{i=0}^{m} b_i x^i = \sum_{i=0}^{2m} \left(\sum_{j=0}^{m} a_j b_{i-j} \right) x^i$$

Aplicación: multiplicar polinomios

Una de las aplicaciones clásicas es multiplicar polinomios rápidamente.

Sean $f = \sum_{i=0}^{m} a_i x^i$, $g = \sum_{i=0}^{m} b_i x^i$ dos polinomios de grado m.

Para multiplicar f y g hay que distribuir:

$$f \cdot g = \sum_{i=0}^{m} a_i x^i \cdot \sum_{i=0}^{m} b_i x^i = \sum_{i=0}^{2m} \left(\sum_{j=0}^{m} a_j b_{i-j} \right) x^i$$

Esto requiere $O(m^2)$ operaciones. Se podrá hacer más eficientemente?

Aplicación: multiplicar polinomios

Una de las aplicaciones clásicas es multiplicar polinomios rápidamente.

Sean $f = \sum_{i=0}^{m} a_i x^i$, $g = \sum_{i=0}^{m} b_i x^i$ dos polinomios de grado m.

Para multiplicar f y g hay que distribuir:

$$f \cdot g = \sum_{i=0}^{m} a_i x^i \cdot \sum_{i=0}^{m} b_i x^i = \sum_{i=0}^{2m} \left(\sum_{j=0}^{m} a_j b_{i-j} \right) x^i$$

Esto requiere $O(m^2)$ operaciones. Se podrá hacer más eficientemente? El producto que damos al anillo de monoide se llama producto de convolución, casualidad?

Aplicación: multiplicar polinomios

Una de las aplicaciones clásicas es multiplicar polinomios rápidamente.

Sean $f = \sum_{i=0}^{m} a_i x^i$, $g = \sum_{i=0}^{m} b_i x^i$ dos polinomios de grado m.

Para multiplicar f y g hay que distribuir:

$$f \cdot g = \sum_{i=0}^{m} a_i x^i \cdot \sum_{i=0}^{m} b_i x^i = \sum_{i=0}^{2m} \left(\sum_{j=0}^{m} a_j b_{i-j} \right) x^i$$

Esto requiere $O(m^2)$ operaciones. Se podrá hacer más eficientemente?
El producto que damos al anillo de monoide se llama producto de convolución, casualidad?
No! La transformada discreta de fourier manda el producto de convolución al producto lugar a lugar!

Aplicación: multiplicar polinomios

Una de las aplicaciones clásicas es multiplicar polinomios rápidamente.

Sean $f = \sum_{i=0}^{m} a_i x^i$, $g = \sum_{i=0}^{m} b_i x^i$ dos polinomios de grado m.

Para multiplicar f y g hay que distribuir:

$$f \cdot g = \sum_{i=0}^{m} a_i x^i \cdot \sum_{i=0}^{m} b_i x^i = \sum_{i=0}^{2m} \left(\sum_{j=0}^{m} a_j b_{i-j} \right) x^i$$

Esto requiere $O(m^2)$ operaciones. Se podrá hacer más eficientemente?
El producto que damos al anillo de monoide se llama producto de convolución, casualidad?
No! La transformada discreta de fourier manda el producto de convolución al producto lugar a lugar!
¿Podremos aprovechar esto para hacer menos cuentas?

FFT: Fast Fourier Transform

Sea $N \geq 2m+1$. Sea $\xi = e^{2\pi i/N}$. Llamamos $\hat{a}_k = \sum_{i=0}^{N-1} a_i \xi^{ki}$.

FFT: Fast Fourier Transform

Sea $N \geq 2m+1$. Sea $\xi = e^{2\pi i/N}$. Llamamos $\hat{a}_k = \sum_{i=0}^{N-1} a_i \xi^{ki}$.

El algoritmo FFT para multiplicar polinomios consiste en:

FFT: Fast Fourier Transform

Sea $N \geq 2m+1$. Sea $\xi = e^{2\pi i/N}$. Llamamos $\hat{a}_k = \sum_{i=0}^{N-1} a_i \xi^{ki}$.

El algoritmo FFT para multiplicar polinomios consiste en:

(0) Tomar N la menor potencia de 2 mayor que $2m+1$. Notar que $N \leq 4m$.

FFT: Fast Fourier Transform

Sea $N \geq 2m+1$. Sea $\xi = e^{2\pi i/N}$. Llamamos $\hat{a}_k = \sum_{i=0}^{N-1} a_i \xi^{ki}$.

El algoritmo FFT para multiplicar polinomios consiste en:

(0) Tomar N la menor potencia de 2 mayor que $2m+1$. Notar que $N \leq 4m$.

(1) Calcular $\hat{a}_k$, $\hat{b}_k$ para $0 \leq k < N$. Son ?? operaciones.

FFT: Fast Fourier Transform

Sea $N \geq 2m+1$. Sea $\xi = e^{2\pi i/N}$. Llamamos $\hat{a}_k = \sum_{i=0}^{N-1} a_i \xi^{ki}$.

El algoritmo FFT para multiplicar polinomios consiste en:

(0) Tomar N la menor potencia de 2 mayor que $2m+1$. Notar que $N \leq 4m$.

(1) Calcular $\hat{a}_k$, $\hat{b}_k$ para $0 \leq k < N$. Son ?? operaciones.

(2) Calcular $\hat{a}_k \cdot \hat{b}_k$ para $0 \leq k < N$. Son N operaciones.

FFT: Fast Fourier Transform

Sea $N \geq 2m+1$. Sea $\xi = e^{2\pi i/N}$. Llamamos $\hat{a}_k = \sum_{i=0}^{N-1} a_i \xi^{ki}$.

El algoritmo FFT para multiplicar polinomios consiste en:

(0) Tomar N la menor potencia de 2 mayor que $2m+1$. Notar que $N \leq 4m$.

(1) Calcular $\hat{a}_k$, $\hat{b}_k$ para $0 \leq k < N$. Son ?? operaciones.

(2) Calcular $\hat{a}_k \cdot \hat{b}_k$ para $0 \leq k < N$. Son N operaciones.

(3) Calcular la transformada de Fourier de los $\hat{a}_k \hat{b}_k$ y así recuperar los coeficientes de fg.

FFT: Fast Fourier Transform

Sea $N \geq 2m+1$. Sea $\xi = e^{2\pi i/N}$. Llamamos $\hat{a}_k = \sum_{i=0}^{N-1} a_i \xi^{ki}$.

El algoritmo FFT para multiplicar polinomios consiste en:

(0) Tomar N la menor potencia de 2 mayor que $2m+1$. Notar que $N \leq 4m$.

(1) Calcular $\hat{a}_k$, $\hat{b}_k$ para $0 \leq k < N$. Son ?? operaciones.

(2) Calcular $\hat{a}_k \cdot \hat{b}_k$ para $0 \leq k < N$. Son N operaciones.

(3) Calcular la transformada de Fourier de los $\hat{a}_k \hat{b}_k$ y así recuperar los coeficientes de fg.

Con un poco de ingenio se puede calcular $\mathcal{F}$ utilizando $O(N \log(N))$ operaciones, para esto se toma N potencia de 2.

FFT: Fast Fourier Transform

Sea $N \geq 2m+1$. Sea $\xi = e^{2\pi i/N}$. Llamamos $\hat{a}_k = \sum_{i=0}^{N-1} a_i \xi^{ki}$.

El algoritmo FFT para multiplicar polinomios consiste en:

(0) Tomar N la menor potencia de 2 mayor que $2m+1$. Notar que $N \leq 4m$.

(1) Calcular $\hat{a}_k$, $\hat{b}_k$ para $0 \leq k < N$. Son ?? operaciones.

(2) Calcular $\hat{a}_k \cdot \hat{b}_k$ para $0 \leq k < N$. Son N operaciones.

(3) Calcular la transformada de Fourier de los $\hat{a}_k \hat{b}_k$ y así recuperar los coeficientes de fg.

Con un poco de ingenio se puede calcular $\mathcal{F}$ utilizando $O(N \log(N))$ operaciones, para esto se toma N potencia de 2.

De este modo se puede multiplicar dos polinomios de grado m utilizando $O(m \log m)$ operaciones con números complejos.

¿Para que sirve esto?

Se utiliza para comprimir imágenes (formato JPEG), audio (MP3) y video (MPEG).