.UBAfiuba
FACULTAD DE INGENIERÍA

# Fast Fourier Transform

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

# Multiplicación de polinomios

## Sean

$A(x)=a_0 + a_1x + a_2x^2 + \ldots + a_dx^d$ polinomio de grado d

$B(x)=b_0 + b_1x + b_2x^2 + \ldots + b_dx^d$ polinomio de grado d

## Queremos calcular

$C(x) = A(x) * B(x)$

# Cálculo “naive”

## El resultado será un nuevo polinomio de grado 2d

$$C(x)=c_0 + c_1x + c_2x^2 + \ldots + c_{2d}x^{2d}$$

## Debemos calcular

2d coeficientes

## Cada coeficiente estará dado por

$$C_k = a_0b_k + a_1b_{k-1} + \ldots + a_kb_0 \qquad (\text{con } a_i = 0,\ b_i=0 \text{ si } i>d)$$

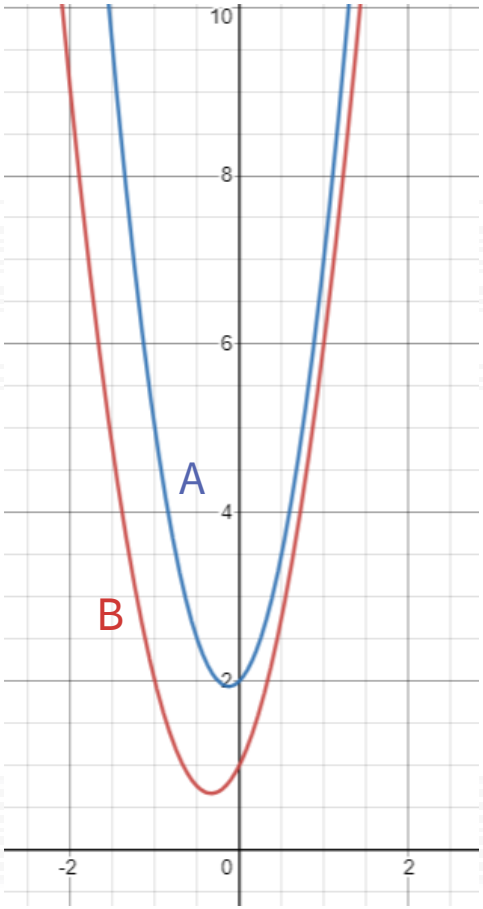
$$c_k = \sum_{i=0}^{k} a_i * b_{k-i}$$

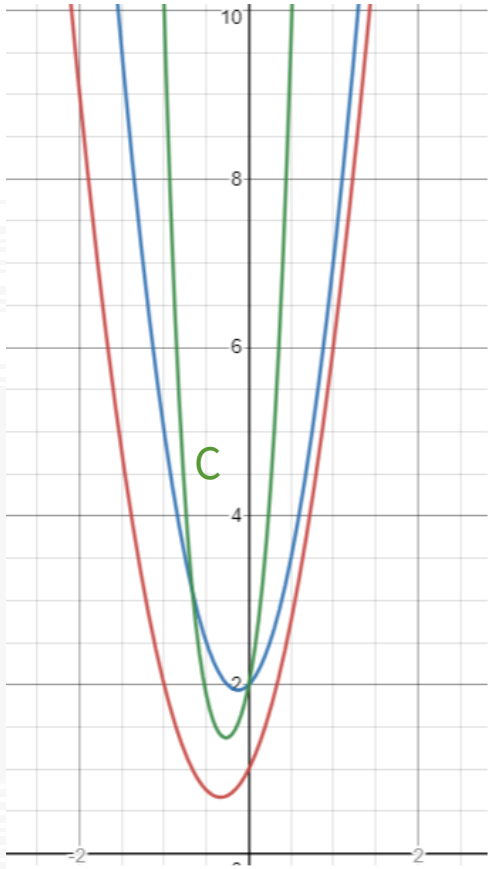
# Ejemplo

## Si

$A(x) = 1 + 2x + 3x^2$

$B(x) = 2 + x + 4x^2$




## Entonces

$A * B = (1 + 2x + 3x^2) * (2 + x + 4x^2)$

$= 1*2 + [(1*x) + (2x * 2)] +$

$[(1*4x^2) + (2x*x) + (2*3x^2)]$

$+ [(2x*4x^2) + (3x^2 *x)] + [3x^2*4x^2]$

$= 2 + 5x + 12x^2 + 11x^3 + 12x^4$

# Análisis de complejidad

## El polinomio resultante

Tendrá 2d coeficientes como máximo.

## Para cada coeficiente

Se deben realizar O(k) multiplicaciones y O(k) sumas

## En total, para un polinomio de grado d

Realizaremos (d+1)*(d+1) multiplicaciones y $d^2$ sumas

## Tenemos una complejidad de $O(d^2)$

¿PODEMOS HACERLO MEJOR?

# Fast Fourier Transform

## El método fue propuesto

A James W. Cooley

Por John W. Tukey en 1963

## Publicaron en 1965 el paper

An Algorithm for the Machine Calculation of Complex Fourier Series

## Luego se supo Gauss propuso un método similar

Aunque por estar escrito en Latín paso desapercibido por muchos años

# Representación por valores

## Un polinomio de grado d

Esta caracterizado unívocamente por el valor de cualquier d+1 puntos distintos

## Ejemplo:

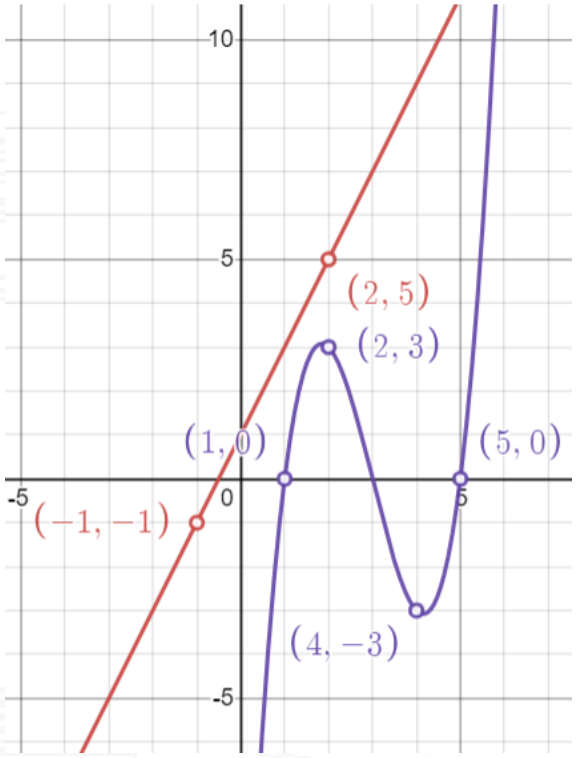
Sean 2 puntos ($x_0$,A($x_0$)) ($x_1$,A($x_1$)). Los mismos definen una única recta

Sean 4 puntos, existe un único polinomio de grado 3 que los contiene

## Podemos elegir

Cualquier set de d+1 puntos diferentes y evaluarlos.

Cada evaluación se puede realizar en O(d) mediante el Algoritmo de Horner




## Representamos un polinomio mediante d+1 puntos

# ¿Son equivalentes las representaciones?

## Partimos de

$$A(x)=a_0 + a_1x + a_2x^2 + \ldots + a_dx^d$$

## Podemos reescribirlos como una multiplicación escalar de 2 vectores

$$A(x) = (1 , x , x^2 \ldots , x^d) * (a_0 , a_1 , a_2 , \ldots , a_d)^t$$

## Si tomamos d+1 números

$$X_0, X_1, \ldots, X_{d+1}$$

# ¿Son equivalentes las representaciones? (cont.)

## Podemos expresarlo como el sistema

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ A(x_2) \\ \dots \\ A(x_d) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^d \\ 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_d & x_d^2 & \dots & x_d^d \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \dots \\ a_d \end{bmatrix}$$

Valores

**M**

(Matriz de Vandermonde)

Coeficientes

# ¿Son equivalentes las representaciones? (cont.)

## Una matriz de Vandermonde

Es invertible si todos los xi son números diferentes

se puede calcular el inverso $M^{-1}$ en $O(d^2)$ !

## Por lo tanto

podemos calcular $M^{-1}$

Y reescribir nuestro sistema

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \dots \\ a_d \end{bmatrix} = M^{-1} * \begin{bmatrix} A(x_0) \\ A(x_1) \\ A(x_2) \\ \dots \\ A(x_d) \end{bmatrix}$$

## Partiendo de lo d+1 puntos

Podemos obtener los coeficientes y reconstruir la expresión polinómica

# Ventajas de la representación por valores

## Usando la representación por coeficientes

Podemos evaluar un $A(x)$ en $O(d)$

Podemos calcular $C(x)=A(x)B(x)$ en $O(d^2)$

## Usando la representación por valores

No podemos calcular directamente $A(x)$ ← debemos interpolar

(a menos que x sea alguno de los puntos $(x_i,A(x_i))$ )

Podemos calcular $C(x)=A(x)B(x)$ en $O(d)$

¿Como?

# Calculo de multiplicación de polinomios en O(d)

## Sean

$A(x)=a_0 + a_1x + a_2x^2 + \ldots + a_dx^d$ polinomio de grado d

$B(x)=b_0 + b_1x + b_2x^2 + \ldots + b_dx^d$ polinomio de grado d

$x_0, x_1, \ldots x_{2d+1}$ 2d+1 valores seleccionados (para polinomio de grado 2d)

## Representamos A y B por valores

$A \rightarrow (x_0 , A(x_0)) (x_1 , A(x_1)) , \ldots , (x_{2d} , A(x_{2d}))$

$B \rightarrow (x_0 , B(x_0)) (x_1 , B(x_1)) , \ldots , (x_{2d} , B(x_{2d}))$

## Calculamos C(x) = A(x)*B(x) como

$(x_0 , A(x_0)*B(x_0)) (x_1 , A(x_1)*B(x_1)) , \ldots , (x_{2d} , A(x_{2d})B(x_{2d}))$

**2d+1 multiplicaciones**

**→ O(d)**

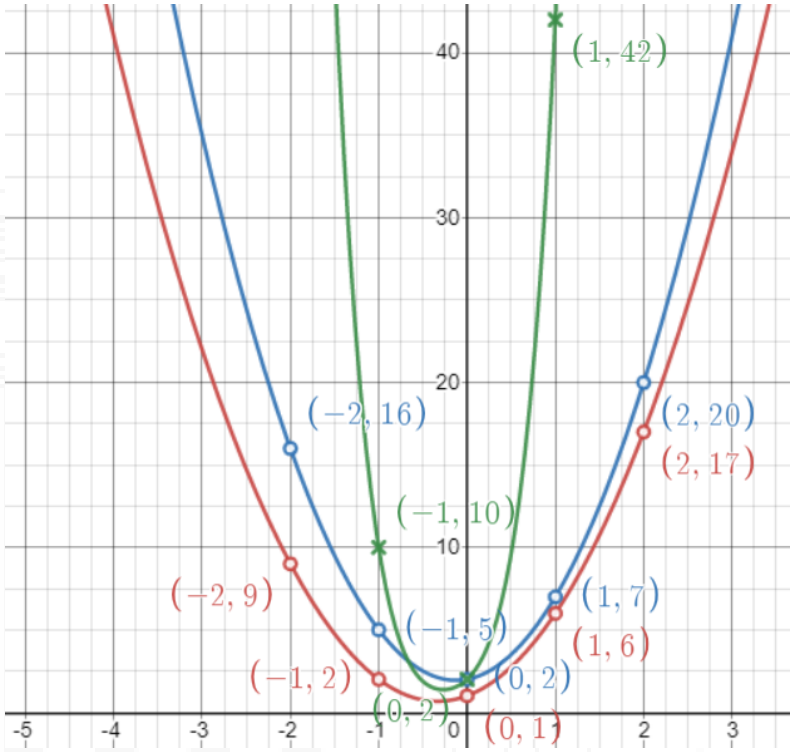
# Ejemplo

## Para

$A(x) = 1 + 2x + 3x^2$

$B(x) = 2 + x + 4x^2$

$X_i = \{-1, 0, 1, -2, 2\}$




## Representamos los polinomios como

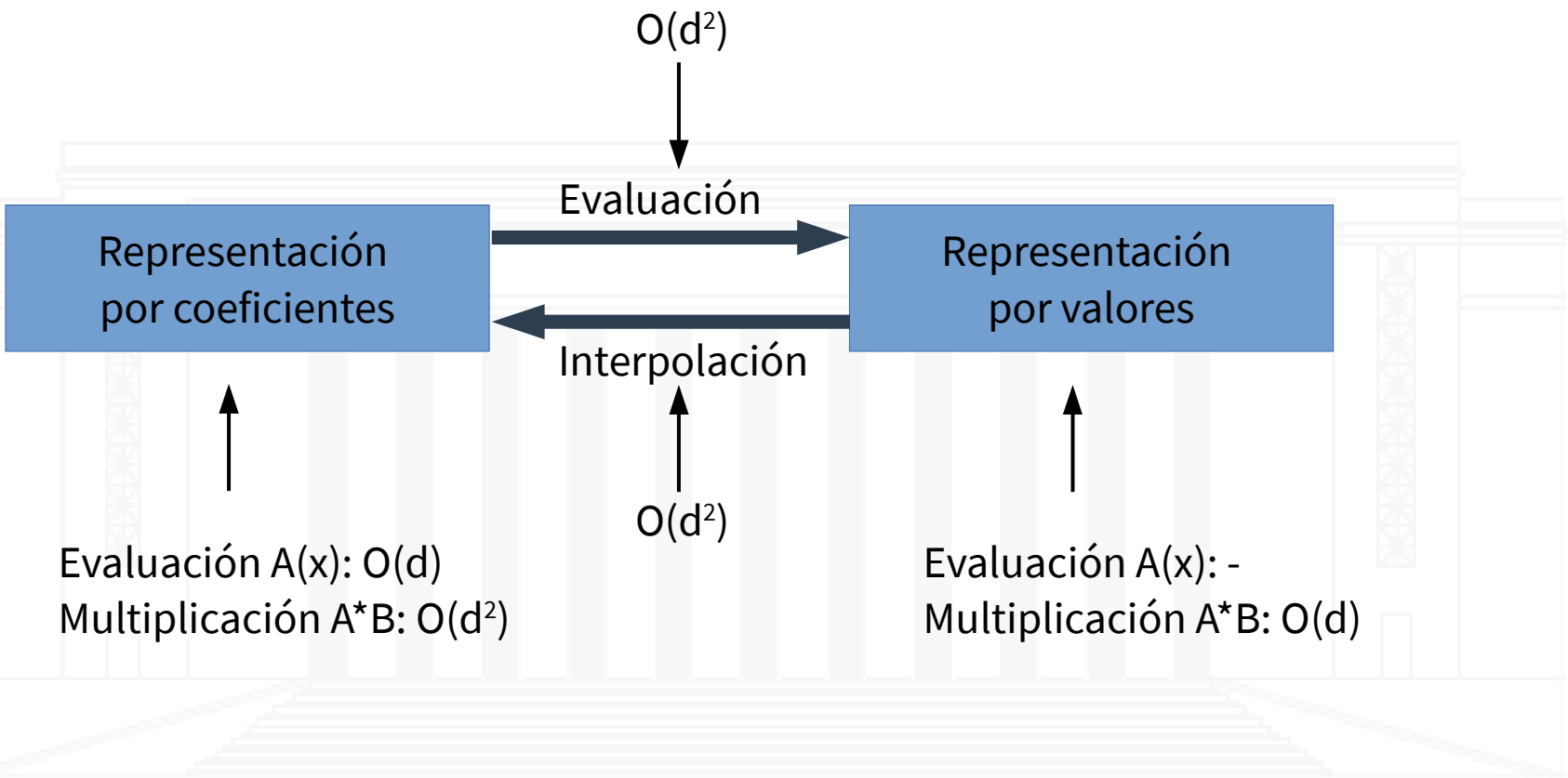
A = (-1, 2) (0, 1) (1, 6) (2, 17) (-2, 9)

B = (-1, 5) (0,2) (1,7) (2 , 20) (-2, 16)

## Calculamos C(x) = A(x)*B(x) como

C = (-1, 2*5 ) (0, 1*2) (1, 6*7) (2, 17*20) (-2, 9*16) = (-1,10) (0,2) (1,42) …

# Sumario


O(d²)
Evaluación
Representación por coeficientes
Representación por valores
Interpolación
O(d²)
Evaluación A(x): O(d)
Multiplicación A*B: O(d²)
Evaluación A(x): -
Multiplicación A*B: O(d)

# Estrategia

## Sean

$A(x)=a_0 + a_1x + a_2x^2 + \ldots + a_dx^d$ polinomio de grado d

$B(x)=b_0 + b_1x + b_2x^2 + \ldots + b_dx^d$ polinomio de grado d

## Para hallar C(x)=A(x)*B(x) C(x) de grado 2d

Seleccionaremos n=2d+1 puntos : $x_0, x_1, \ldots x_{n-1}$

Evaluaremos A(xi) y B(xi) para 0≤i<n ← $O(n^2)$

Multiplicaremos C(xi) = A(xi) * B(xi) para 0≤i<n ← $O(n)$

Por interpolación obtendremos C(X) ← $O(n^2)$

**Queremos Mejorar!!!**

# Selección conveniente de puntos

## Una elección conveniente de puntos

Puede simplificar considerablemente los cálculos de la evaluacion de A(x)

## Si elegimos el par de puntos $x_i$ y $-x_i$

Entonces el computo de $A(x_i)$ y $A(-x_i)$ comparten cálculos

## Todas las potencias pares de de $x_i$

coinciden con las de $-x_i$

# Analicemos...

## Para

$A(x)=a_0 + a_1x + a_2x^2 + \ldots + a_dx^d$

(suponemos que d es par. Igualmente se puede desarrollar para impar)

## Podemos expresarlo separando los exponentes pares e impares

$A(x)=( a_0 + a_2x^2 + \ldots + a_dx^d ) + ( a_1x + a_3x^3 + \ldots + a_{d-1}x^{d-1} )$

## Y luego agrupando por x la segunda parte

$A(x)=( a_0 + a_2x^2 + \ldots + a_dx^d ) + x( a_1 + a_3x^2 + \ldots + a_{d-1}x^{d-2} )$

$A_p(x^2)$ $A_i(x^2)$

# Reducción de los cálculos

## Reescribimos

$$A(x) = A_p(x^2) + xA_i(x^2)$$

## Si tomamos $x_i$ y $-x_i$

$$A(x_i) = A_p(x_i^2) + x_iA_i(x_i^2)$$

$$A(-x_i) = A_p(x_i^2) - x_iA_i(x_i^2)$$

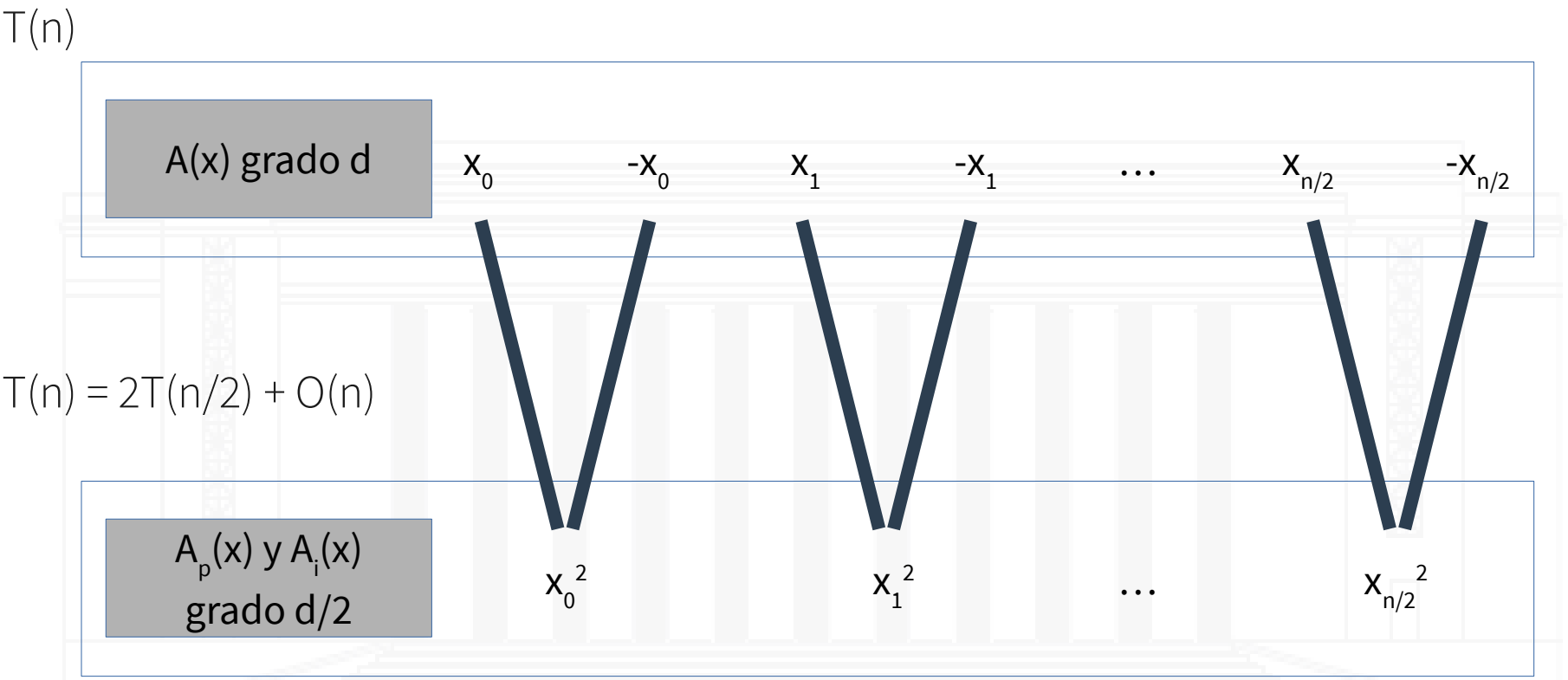
## Es decir que si utilizamos pares de puntos $\pm x_0, \ldots, \pm x_{n/2}$

Reducimos la cantidad de cálculos de evaluación a la mitad

## Y Para cada evaluación dividimos el problema

al cálculo de $A_p$ y $A_i$ (que son polinomios de grado d/2)

# División y conquista




**… pero esta recursión no funciona.**

No tengo pares de valores positivos – negativos para unir … excepto si….

# “Si la solución no es Real... es Compleja”

## Necesitamos que en cada subproblema

Existan pares de números de signo contrario

## Podremos lograrlo

Usando números complejos.

## Utilizaremos para nuestros puntos

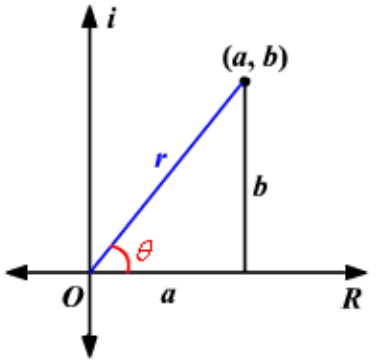
n raíces complejas de la unidad

Que dan solución a la ecuación $z^n=1$

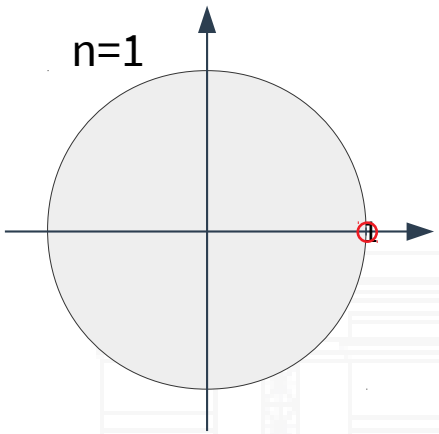
**Números Complejos**

- Exp. binómica: a+bi
- Exp. Trigonométrica: r*(cos θ + i sen θ)

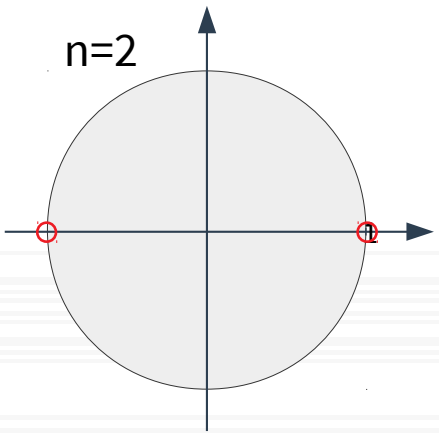
  r = sqrt ($a^2 + b^2$)
  θ = arcotg (b/a)
- Exp. exponencial: $r\, e^{i\theta}$



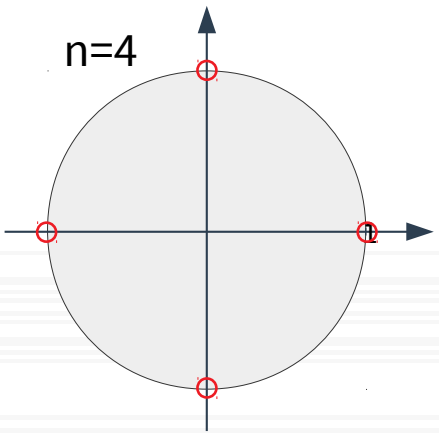
# “Si la solución no es Real... es Compleja” (cont.)



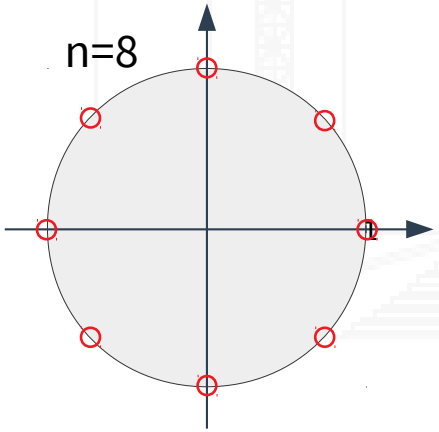

$(1,0)^1 = 1$



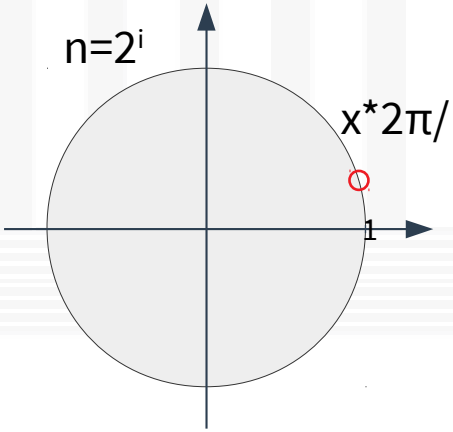

$(1,2\pi)^2 = 1$
$(1,\pi)^2 = 1$




$(1,0)^4 = 1$
$(1,\pi/2)^4 = 1$
$(1,\pi)^4 = 1$
$(1,-\pi/2)^4 = 1$




...




Con x de 0 a n-1

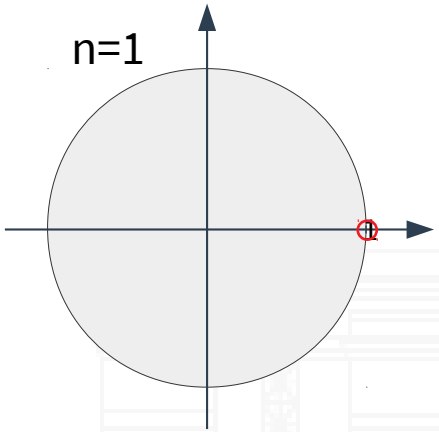
**Multiplicación compleja**

$$(r_1,\theta_1) \times (r_2,\theta_2) = (r_1 r_2, \theta_1 + \theta_2)$$

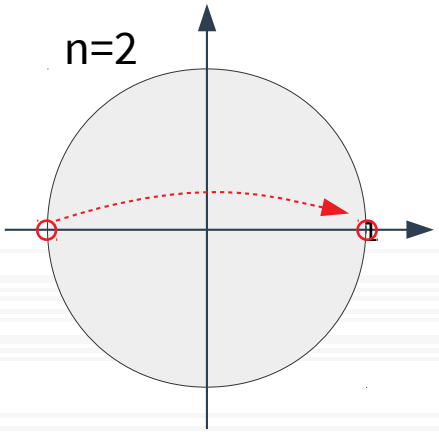
- Si r=1

  $z^n=(1, n\theta)$

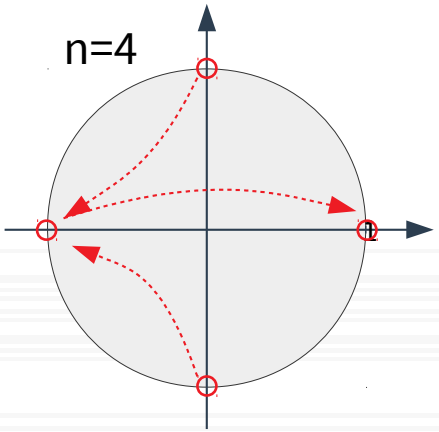
# “Si la solución no es Real... es Compleja” (cont.)




$(1,0)^2 = 1$

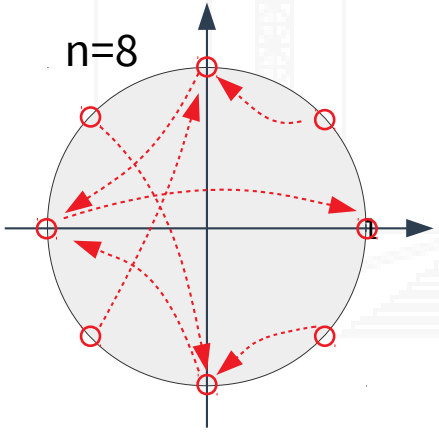



$(1,2\pi)^2 = 1$
$(1,\pi)^2 = 1$




$(1,0)^2 = 1$
$(1,\pi/2)^2 = -1$

$(1,\pi)^2 = 1$
$(1,-\pi/2)^2 = -1$

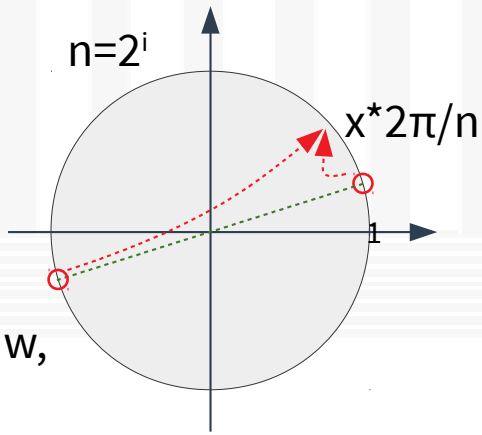



En cada paso me quedo con la mitad de puntos

…

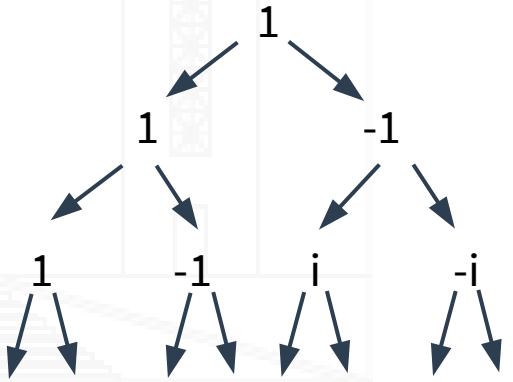
**Llamaremos $w = e^{2\pi/n}$**

Las n-esimas raices de w, podemos escribirlas como $w^0, w^2, \ldots, w^{n-1}$




En nuestro planteo para cada subproblema elevamos al cuadrado los puntos

$$A(x) = A_p(x^2) + xA_i(x^2)$$



# Pseudocódigo – Fast Fourier Transform

## Entrada:

A(x) polinomio de grado ≤ n-1 (n potencia de 2)

n-esimas raices de la unidad.

```
FFT(A,n)

Si n=1
     retornar A(1) // evaluo A con punto 1

Expresar A(X) de la forma A_p(x²) + x*A_i(x²)

w=e^(2π/n)

R_p = FFT(A_p, n/2)
R_i = FFT(A_i, n/2)

Sea R vector de n posiciones
Desde j=0 a n/2 -1
     R[j] = R_p[j] + w^j* R_i[j]
     R[j + n/2] = R_p[j] - w^j*R_i[j]

Retornar R
```

# Complejidad

## Cada subproblema

Se divide en 2 subproblemas de la mitad de puntos (w)

## En cada subproblema

Se realizan O(n) cálculos

## T(n) = 2T(n/2) + O(n)

Por teorema maestro tenemos un complejidad de O(nlogn)

```
FFT(A,n)

Si n=1
     retornar A(1) // evaluo A con punto 1

Expresar A(X) de la forma A_p(x^2) + x*A_i(x^2)

w=e^(2π/n)

R_p = FFT(A_p, n/2)
R_i = FFT(A_i, n/2)

Sea R vector de n posiciones
Desde j=0 a n/2 -1
     R[j] = R_p[j] + w^j* R_i[j]
     R[j + n/2] = R_p[j] - w^j*R_i[j]

Retornar R
```

# Estrategia recargada

## Sean

$A(x)=a_0 + a_1x + a_2x^2 + \ldots + a_dx^d$ polinomio de grado d

$B(x)=b_0 + b_1x + b_2x^2 + \ldots + b_dx^d$ polinomio de grado d

## Para hallar C(x)=A(x)*B(x)

Seleccionaremos n=2d+1 puntos : $x_0, x_1, \ldots x_{n-1}$

Evaluaremos A(xi) y B(xi) para 0≤i<n ← O(nlogn)

Multiplicaremos C(xi) = A(xi) * B(xi) para 0≤i<n ← O(n)

Por interpolación obtendremos C(X) ← O(??)

**Pendiente!!!**

# Interpolación – DFT Matrix

## Partimos de la expresión

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ A(x_2) \\ \dots \\ A(x_{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \dots \\ a_{n-1} \end{bmatrix}$$

## Para la resolución utilizamos $x_k = w^k$

Matriz discreta de transformada de Fourier (DFT)

$$\begin{bmatrix} A(w^0) \\ A(w^1) \\ A(w^2) \\ \dots \\ A(w^{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w^1 & w^2 & \dots & w^{n-1} \\ 1 & w^2 & w^4 & \dots & w^{2(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & w^{n-1} & w^{2(n-1)} & \dots & w^{(n-1)(n-1)} \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \dots \\ a_{n-1} \end{bmatrix}$$

# Interpolación – Matriz inversa

## Queremos operar de forma eficiente con $M^{-1}$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \dots \\ a_{n-1} \end{bmatrix} = M^{-1} * \begin{bmatrix} A(w^0) \\ A(w^1) \\ A(w^2) \\ \dots \\ A(w^{n-1}) \end{bmatrix}$$

## Invirtiendo M podemos obtener

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \dots \\ a_{n-1} \end{bmatrix} = 1/n \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w^{-1} & w^{-2} & \dots & w^{-(n-1)} \\ 1 & w^{-2} & w^{-4} & \dots & w^{-2(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & w^{-(n-1)} & w^{-2(n-1)} & \dots & w^{-(n-1)(n-1)} \end{bmatrix} * \begin{bmatrix} A(w^0) \\ A(w^1) \\ A(w^2) \\ \dots \\ A(w^{n-1}) \end{bmatrix}$$

Similar a la matriz de evaluación (reemplazando cada w por $w^{-1}$)

Siguen siendo raíces de la unidad

# Inverse Fast Fourier Transform

## Al ser también raíces de la unidad

Podemos basarnos en el mismo principio que FFT con pocos cambios

## Lo llamaremos IFFT

Recibirá por parámetro la representación por valor

## Obtendrá en O(nlogn)

La representación por coeficientes

# Interpolación

## Entrada:

P: n valores (n potencia de 2)

Con $P_i = A(w^i)$

n-esimas raíces de la unidad a utilizar.

## Complejidad de O(nlogn)

Por teorema maestro

```
IFFT(P,n)

Si n=1
     retornar P

P_p puntos pares
P_i puntos impares

W=(1/n)*e^(-2π/n)

R_p = IFFT(A_p, n/2)
R_i = IFFT(A_i, n/2)

Sea R vector de n posiciones
Desde j=0 a n/2 -1
     R[j] = R_p[j] + w^j* R_i[j]
     R[j + n/2] = R_p[j] - w^j*R_i[j]

Retornar R
```

# Estrategia Final

## Sean

$A(x)=a_0 + a_1x + a_2x^2 + \ldots + a_dx^d$ polinomio de grado d

$B(x)=b_0 + b_1x + b_2x^2 + \ldots + b_dx^d$ polinomio de grado d

## Para hallar C(x)=A(x)*B(x)

Seleccionaremos $n=2d+1$ puntos : $x_0, x_1, \ldots x_{n-1}$

Evaluaremos A(xi) y B(xi) para $0 \leq i < n$ ← O(nlogn)

Multiplicaremos C(xi) = A(xi) * B(xi) para $0 \leq i < n$ ← O(n)

Por interpolación obtendremos C(X) ← O(nlogn)

**O(nlogn)**

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Marzo de 2021