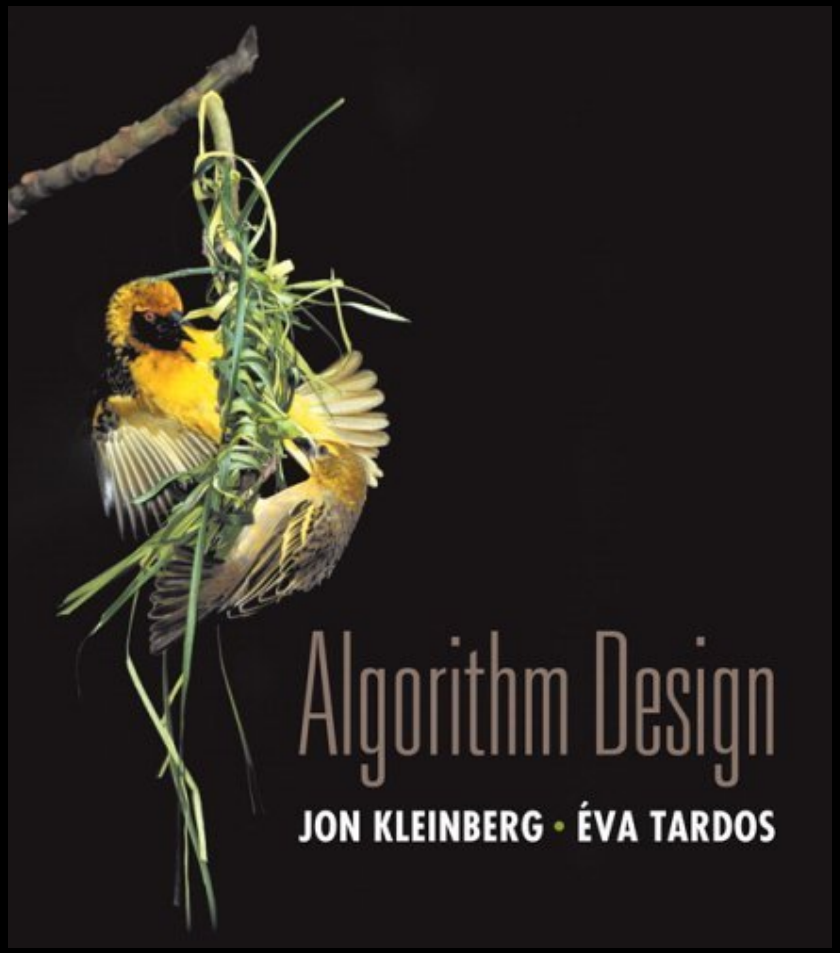


Chapitre 5

Diviser pour régner (suite)

Convolution et FFT

FFT = Fast Fourier Transform
(Transformée de Fourier rapide)

FFT : Applications

Applications.

- Optique, acoustique, physique quantique, télécommunications, systèmes de contrôle, traitement du signal, reconnaissance de la parole, compression de données, traitement des images.
- Lecture DVD, JPEG, MP3, MRI, CAT.
- Solutions numériques de l'équation de Poisson.

> The FFT is one of the truly great computational developments of this [20th] century. It has changed the face of science and engineering so much that it is not an exaggeration to say that life as we know it would be very different without the FFT. *-Charles van Loan*

FFT : un peu d'histoire

Gauss (1805, 1866). Analyse du mouvement périodique de Céres.

Runge-König (1924). Fondations théoriques.

Danielson-Lanczos (1942). Algorithme efficace.

Cooley-Tukey (1965). Utilisé pour le contrôle des tests nucléaires en URSS et pour le pistage des sous-marins. FFT redécouvert et rendu populaire.

L' importance de FFT ne fut pleinement reconnue qu'avec l'utilisation massive des ordinateurs.

Polynômes : représentation par coefficients

Polynôme. [représentation par coefficients]

$$A(x) = a_0 + a_1 x + a_2 x^2 + L + a_{n-1} x^{n-1}$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + L + b_{n-1} x^{n-1}$$

Addition : $O(n)$ opérations arithmétiques.

$$A(x) + B(x) = (a_0 + b_0) + (a_1 + b_1) x + L + (a_{n-1} + b_{n-1}) x^{n-1}$$

Evaluation : $O(n)$ opérations arithmétiques (méthode de Horner.)

$$A(x) = a_0 + (x(a_1 + x(a_2 + L + x(a_{n-2} + x(a_{n-1}))L))$$

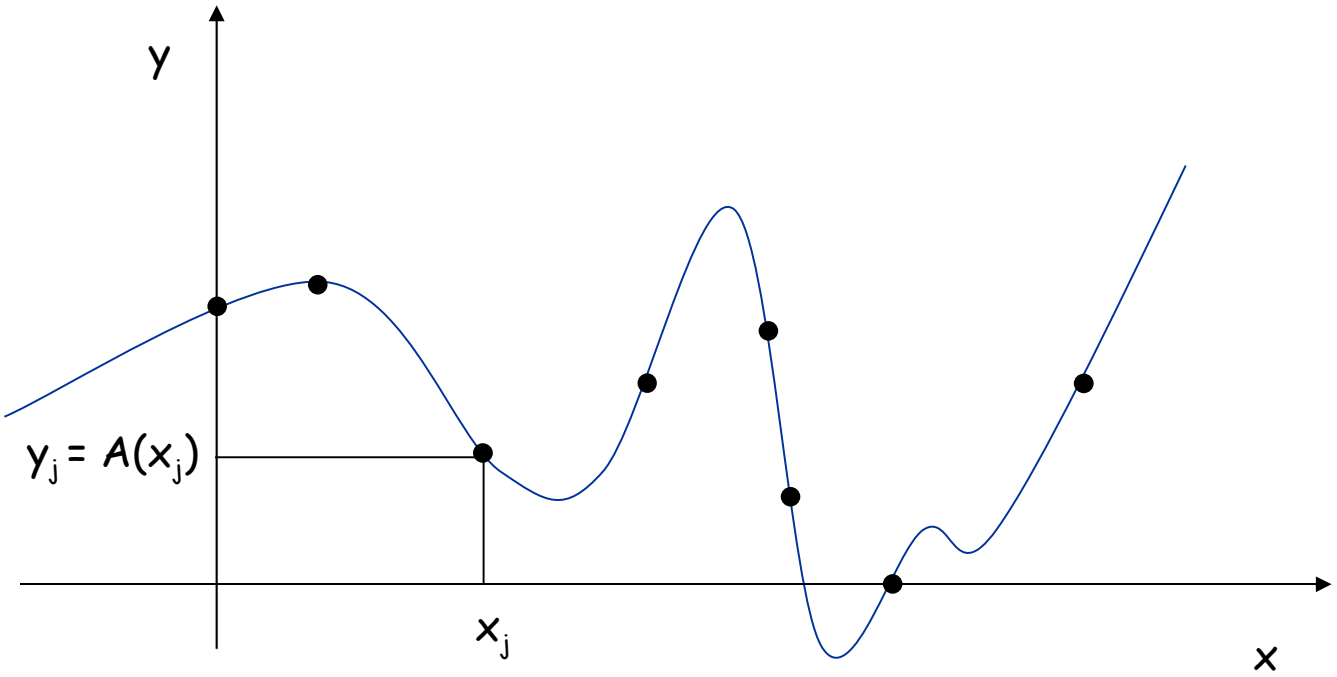
Multiplication (convolution) : $O(n^2)$ en utilisant la méthode brutale.

$$A(x) \times B(x) = \sum_{i=0}^{2n-2} c_i x^i, \text{ where } c_i = \sum_{j=0}^{i} a_j b_{i-j}$$

Polynômes : représentation par couples (point,valeur)

Théorème fondamental de l'algèbre. [Gauss] Un polynôme de degré n à coefficients complexes a n racines complexes (avec multiplicités.)

Corollaire. Un polynôme de degré n-1 A(x) est entièrement déterminé par son évaluation en n valeurs distinctes de x.



Polynômes : représentation par couples (point,valeur)

Polynôme. [représentation points-valeurs]

$$A(x): (x_0, y_0), K, (x_{n-1}, y_{n-1})$$
$$B(x): (x_0, z_0), K, (x_{n-1}, z_{n-1})$$

Addition : $O(n)$ opérations arithmétiques.

$$A(x)+B(x): \ (x_0, y_0+z_0), K, (x_{n-1}, y_{n-1}+z_{n-1})$$

Multiplication : $O(n)$, mais $2n-1$ points sont nécessaires.

$$A(x) \times B(x): \ (x_0, y_0 \times z_0), K, (x_{2n-1}, y_{2n-1} \times z_{2n-1})$$

Evaluation : $O(n^2)$, en utilisant la formule de Lagrange.

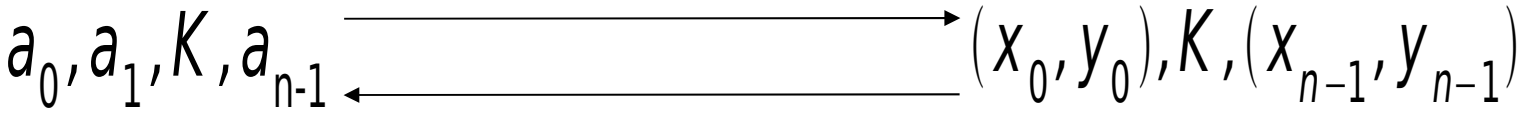
$$A(x) = \sum_{k=0}^{n-1} y_k \frac{\prod_{j \neq k} (x - x_j)}{\prod_{j \neq k} (x_k - x_j)}$$

Passage d'une représentation à l'autre

Compromis. Évaluation rapide ou multiplication rapide. On veut les deux !

Représentation	Multiplication	Evaluation
Coefficients	$O(n^2)$	$O(n)$
Points-valeurs	$O(n)$	$O(n^2)$

But. Rendre toutes les opérations rapides en permettant des conversions efficaces entre les deux représentations.



représentation par coefficients

représentation par points-valeurs

Passage d'une représentation à l'autre : méthode brutale

Coefficients vers Points-valeurs. Étant donné un polynôme $a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$, l'évaluer en n points distincts $x_0, \ldots, x_{n-1}$.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ M \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & L & x_0^{n-1} \\ 1 & x_1 & x_1^2 & L & x_1^{n-1} \\ 1 & x_2 & x_2^2 & L & x_2^{n-1} \\ M & M & M & O & M \\ 1 & x_{n-1} & x_{n-1}^2 & L & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ M \\ a_{n-1} \end{bmatrix}$$

$O(n^2)$ pour la multiplication matrice x vecteur

$O(n^3)$ pour l'élimination de Gauss

la matrice de Vandermonde est inversible ssi les x_i sont deux à deux distincts

Points-valeurs vers Coefficients. Étant donnés n points distincts $x_0, \ldots, x_{n-1}$ et n valeurs $y_0, \ldots, y_{n-1}$, trouver l'unique polynôme $A(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$ tel que $A(x_i) = y_i$, pour $i = 1, \ldots, n-1$.

Coefficient vers points-valeurs : intuition

Coefficients vers Points-valeurs. Étant donné un polynôme $a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$, l'évaluer en n points distincts $x_0, \ldots, x_{n-1}$.

Divide. Casser le polynôme selon la parité des puissances.

- $A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7$.
- $A_{even}(x) = a_0 + a_2 x + a_4 x^2 + a_6 x^3$.
- $A_{odd}(x) = a_1 + a_3 x + a_5 x^2 + a_7 x^3$.
- $A(x) = A_{even}(x^2) + x\, A_{odd}(x^2)$.
- $A(-x) = A_{even}(x^2) - x\, A_{odd}(x^2)$.

Intuition. On choisit les deux points ± 1.

- $A(1) = A_{even}(1) + 1\, A_{odd}(1)$.
- $A(-1) = A_{even}(1) - 1\, A_{odd}(1)$.

Pour évaluer un polynôme de degré $\leq n$ en 2 points, il suffit d'évaluer deux polynômes de degré $\leq \frac{1}{2}n$ en 1 point.

Coefficient vers points-valeurs : intuition

Coefficients vers Points-valeurs. Étant donné un polynôme $a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$, l'évaluer en n points distincts $x_0, \ldots, x_{n-1}$.

Divide. Casser le polynôme selon la parité des puissances.

- $A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7$.
- $A_{even}(x) = a_0 + a_2 x + a_4 x^2 + a_6 x^3$.
- $A_{odd}(x) = a_1 + a_3 x + a_5 x^2 + a_7 x^3$.
- $A(x) = A_{even}(x^2) + x\, A_{odd}(x^2)$.
- $A(-x) = A_{even}(x^2) - x\, A_{odd}(x^2)$.

Intuition. On choisit les quatre points $\pm 1, \pm i$.

- $A(1) = A_{even}(1) + 1\, A_{odd}(1)$.
- $A(-1) = A_{even}(1) - 1\, A_{odd}(1)$.
- $A(i) = A_{even}(-1) + i\, A_{odd}(-1)$.
- $A(-i) = A_{even}(-1) - i\, A_{odd}(-1)$.

Pour évaluer un polynôme de degré $\leq n$ en 4 points, il suffit d'évaluer deux polynômess de degré $\leq \frac{1}{2}n$ en 2 points.

Transformée de Fourier discrète

Coefficients vers Points-valeurs. Étant donné un polynôme $a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$, l'évaluer en n points distincts $x_0, \ldots, x_{n-1}$.

Idée clé : choisir $x_k = \omega^k$ où ω est la n^e racine principale de l'unité.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ M \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & L & 1 \\ 1 & w^1 & w^2 & w^3 & L & w^{n-1} \\ 1 & w^2 & w^4 & w^6 & L & w^{2(n-1)} \\ 1 & w^3 & w^6 & w^9 & L & w^{3(n-1)} \\ M & M & M & M & O & M \\ 1 & w^{n-1} & w^{2(n-1)} & w^{3(n-1)} & L & w^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ M \\ a_{n-1} \end{bmatrix}$$

Transformée de Fourier discrète — Matrice de Fourier F_n

Rappel sur les racines de l'unité

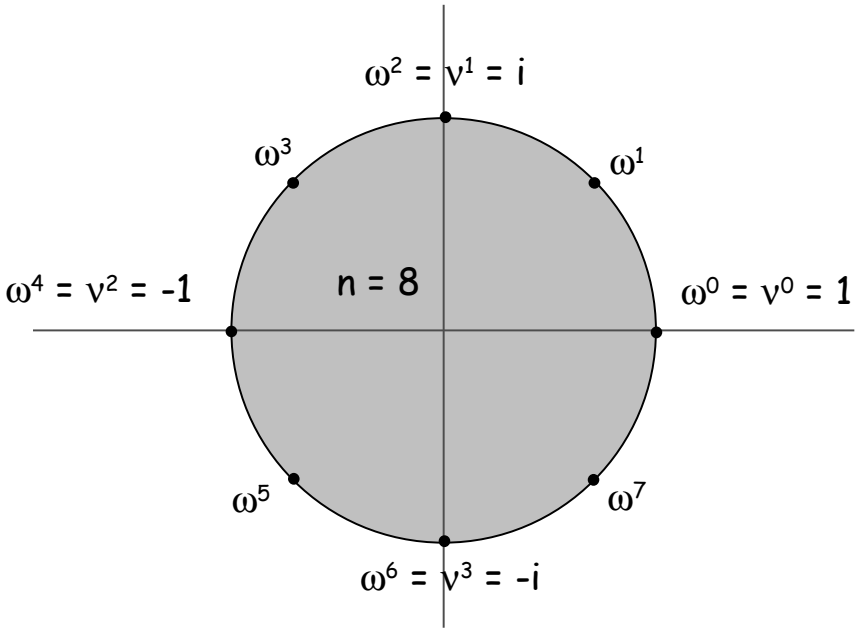
Déf. Une n^e racine de l'unité est un nombre complexe x tel que $x^n = 1$.

Fait. Les n^e racines de l' unité sont : $\omega^0, \omega^1, \ldots, \omega^{n-1}$ où $\omega = e^{2\pi i / n}$.

Preuve. $(\omega^k)^n = (e^{2\pi i k / n})^n = (e^{\pi i})^{2k} = (-1)^{2k} = 1$.

Fait. Les $\frac{1}{2}n^e$ racines de l' unité sont : $\nu^0, \nu^1, \ldots, \nu^{n/2-1}$ où $\nu = e^{4\pi i / n}$.

Fait. $\omega^2 = \nu$ et $(\omega^2)^k = \nu^k$.



Transformée de Fourier rapide

But. Evaluer un polynôme de degré n-1 $A(x) = a_0 + \ldots + a_{n-1}\, x^{n-1}$ en les n^e racines de l'unité : $\omega^0, \omega^1, \ldots, \omega^{n-1}$.

Divide. Casser le polynôme selon la parité des puissances.

- $A_{even}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n/2-2}\, x^{(n-1)/2}$.
- $A_{odd}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n/2-1}\, x^{(n-1)/2}$.
- $A(x) = A_{even}(x^2) + x\, A_{odd}(x^2)$.

Conquer. Évaluer $A_{even}(x)$ et $A_{odd}(x)$ en les $\frac{1}{2}n^e$ racines de l'unité : $\nu^0, \nu^1, \ldots, \nu^{n/2-1}$.

Combiner.

- $A(\omega^k) = A_{even}(\nu^k) + \omega^k\, A_{odd}(\nu^k), \quad 0 \le k < n/2$
- $A(\omega^{k+n}) = A_{even}(\nu^k) - \omega^k\, A_{odd}(\nu^k), \quad 0 \le k < n/2$

$\omega^{k+n} = -\omega^k$

$\nu^k = (\omega^k)^2 = (\omega^{k+n})^2$

FFT : l'algorithme

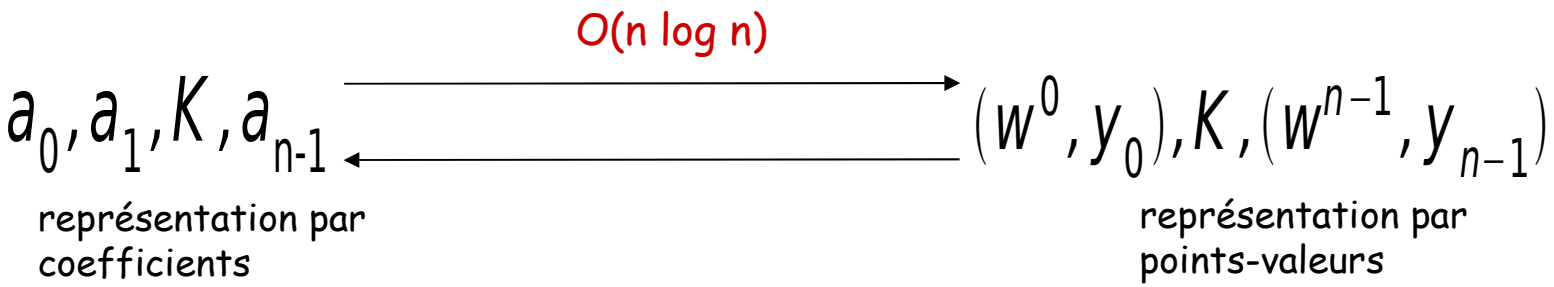
```
fft(n, a_0,a_1,…,a_n-1) {
    si (n == 1) retourner a_0

    (e_0,e_1,…,e_n/2-1) ← FFT(n/2, a_0,a_2,a_4,…,a_n-2)
    (d_0,d_1,…,d_n/2-1) ← FFT(n/2, a_1,a_3,a_5,…,a_n-1)

    pour k = 0 à n/2 - 1 {
        ω^k ← e^2πik/n
        y_k     ← e_k + ω^k d_k
        y_k+n/2 ← e_k - ω^k d_k
    }

    retourner (y_0,y_1,…,y_n-1)
}
```

FFT : résumé

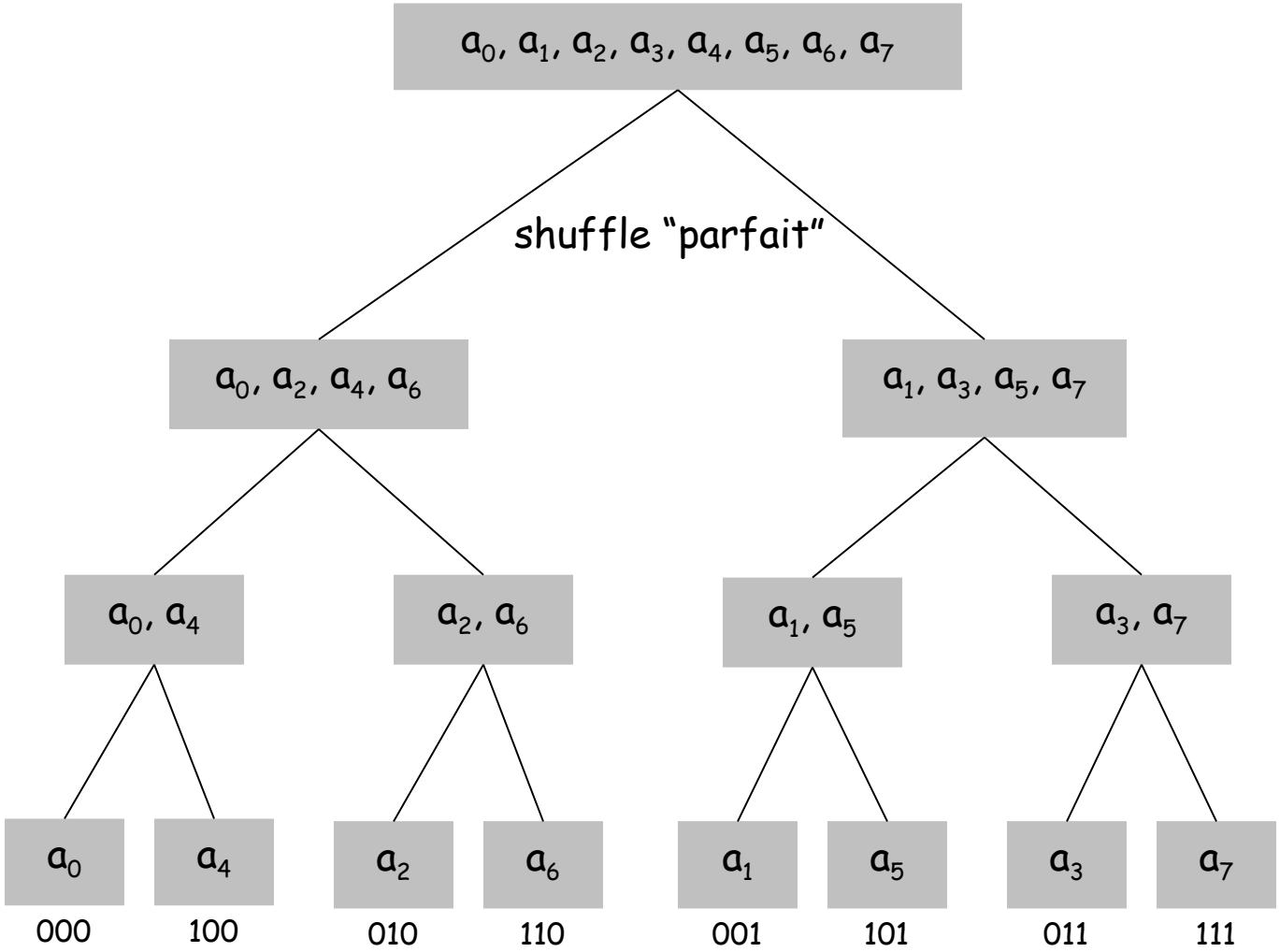
on suppose que n est une puissance de 2

Théorème. L'algorithme FFT évalue un polynôme de degré n-1 en chacune des n^e racines de l'unité, en $O(n \log n)$ étapes de calcul.

Temps d'exécution. $T(2n) = 2T(n) + O(n) \Rightarrow T(n) = O(n \log n)$.



Arbre de récursion



ordre "bit-reversed"

Points-valeurs vers Coefficients : transformée de Fourier discrète inverse

But. Étant données les valeurs $y_0, \ldots, y_{n-1}$, trouver l'unique polynôme $A(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1}$ tel que $A(\omega^i) = y_i$, i = 0, …, n-1.

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ M \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & L & 1 \\ 1 & w^1 & w^2 & w^3 & L & w^{n-1} \\ 1 & w^2 & w^4 & w^6 & L & w^{2(n-1)} \\ 1 & w^3 & w^6 & w^9 & L & w^{3(n-1)} \\ M & M & M & M & O & M \\ 1 & w^{n-1} & w^{2(n-1)} & w^{3(n-1)} & L & w^{(n-1)(n-1)} \end{bmatrix}^{-1} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ M \\ y_{n-1} \end{bmatrix}$$

DFT (transformé de Fourier discrète) inverse

Matrice de Fourier inverse $(F_n)^{-1}$

FFT inverse

Propriété. La matrice de Fourier inverse est donnée par la formule suivante :

$$G_n = \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & 1 & L & 1 \\ 1 & w^{-1} & w^{-2} & w^{-3} & L & w^{-(n-1)} \\ 1 & w^{-2} & w^{-4} & w^{-6} & L & w^{-2(n-1)} \\ 1 & w^{-3} & w^{-6} & w^{-9} & L & w^{-3(n-1)} \\ M & M & M & M & O & M \\ 1 & w^{-(n-1)} & w^{-2(n-1)} & w^{-3(n-1)} & L & w^{-(n-1)(n-1)} \end{bmatrix}$$

Conséquence. Pour calculer la transformée de Fourier inverse, on reprend l'algorithme FFT en utilisant $\omega^{-1} = e^{-2\pi i / n}$ comme n^e racine principale de l'unité (et on divise par n).

FFT inverse : preuve de correction

Proposition. F_n et G_n sont inverses l'une de l'autre.

Preuve.

$$\left(F_n G_n\right)_{kk'} = \frac{1}{n}\sum_{j=0}^{n-1} w^{kj} w^{-jk'} \;=\; \frac{1}{n}\sum_{j=0}^{n-1} w^{(k-k')j} = \uparrow \begin{cases} 1 & \text{if } k = k' \\ 0 & \text{otherwise} \end{cases}$$

cf. lemme de sommation ci-dessous

Lemme de sommation. Soit ω une n^e racine primitive de l'unité. Alors

$$\sum_{j=0}^{n-1} w^{kj} = \begin{cases} n & \text{if } k \equiv 0 \bmod n \\ 0 & \text{otherwise} \end{cases}$$

Preuve.

- Si k est un multiple de n, alors $\omega^k = 1 \;\Rightarrow$ la somme vaut n.
- Chaque ω^k est une racine de $x^n - 1 = (x - 1)(1 + x + x^2 + \ldots + x^{n-1})$.
- Si $\omega^k \neq 1$ on a : $1 + \omega^k + \omega^{k(2)} + \ldots + \omega^{k(n-1)} = 0 \;\Rightarrow$ la somme 0. ▪

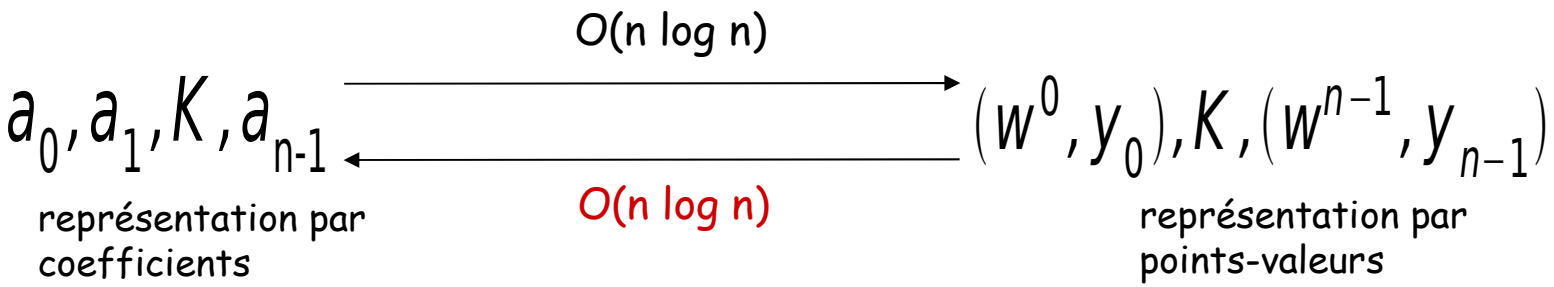
FFT inverse : l'algorithme

```
ffti(n, a_0,a_1,…,a_{n-1}) {
    si (n == 1) retourner a_0

    (e_0,e_1,…,e_{n/2-1}) ← FFT(n/2, a_0,a_2,a_4,…,a_{n-2})
    (d_0,d_1,…,d_{n/2-1}) ← FFT(n/2, a_1,a_3,a_5,…,a_{n-1})

    pour k = 0 à n/2 - 1 {
        ω^k ← e^{-2πik/n}
        y_k     ← (e_k + ω^k d_k) / n
        y_{k+n/2} ← (e_k - ω^k d_k) / n
    }

    retourner (y_0,y_1,…,y_{n-1})
}
```

FFT inverse : résumé

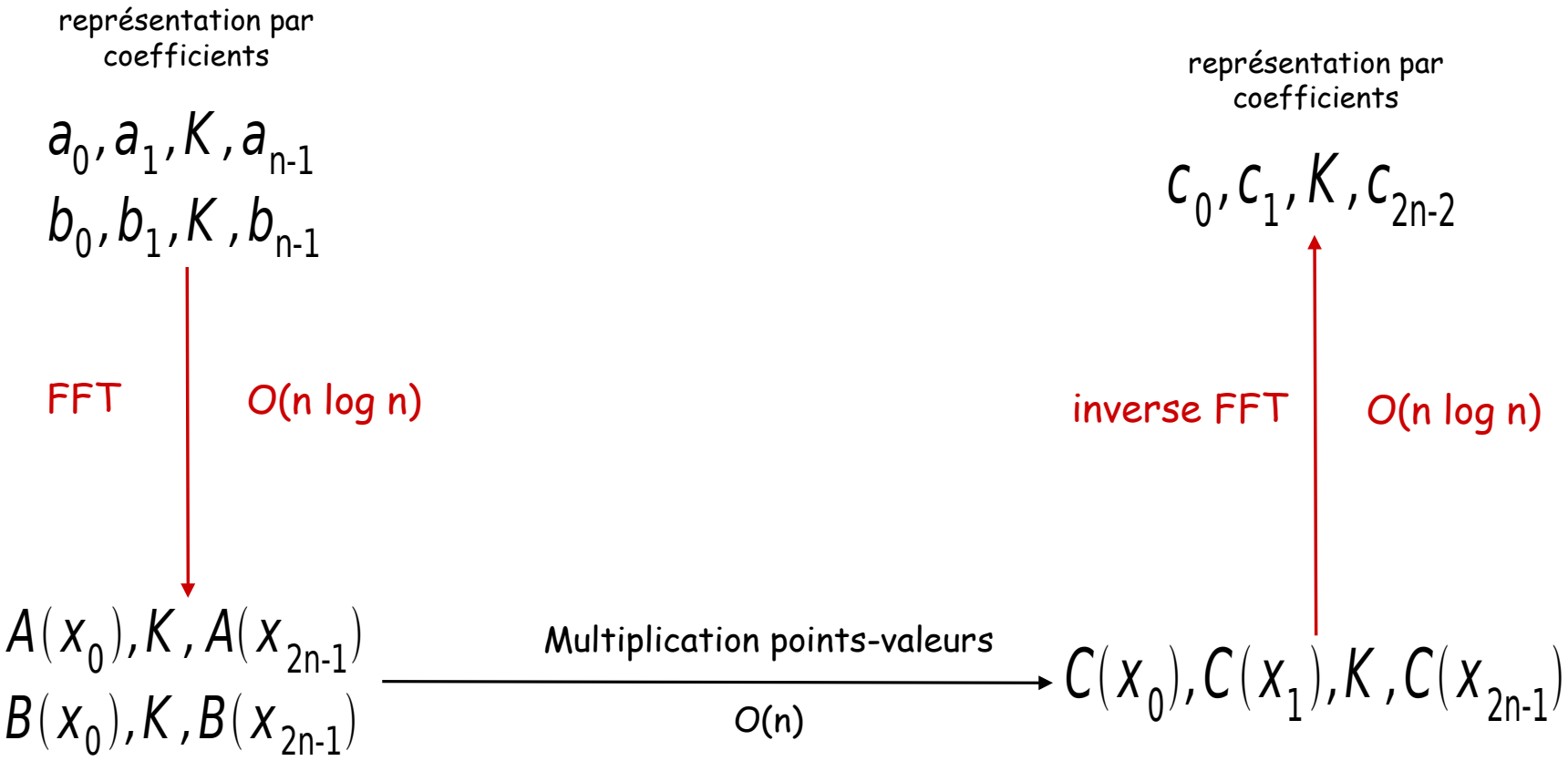
Théorème. L'algorithme FFT inverse interpole un polynôme de degré $\leq$ n-1 étant données les valeurs prises en chacune des n^e racines de l'unité, en $O(n \log n)$ étapes de calcul.

on suppose que n est une puissance de 2



Multiplication de polynômes

Théorème. On multiplier deux polynômes de degré n-1 en $O(n \log n)$ étapes de calcul.



FFT en pratique

"Fastest Fourier transform in the West." [Frigo & Johnson]

- Bibliothèque C optimisée.
- Contient : DFT, DCT, réel, complexe, en dimension quelconque.
- Prix Wilkinson 1999 du logiciel numérique.
- Portable et compétitif.

Détails d'implémentation.

- Au lieu d'exécuter un algorithme prédéterminé, il évalue le matériel et utilise un compilateur dédié pour engendrer un code optimisé adapté à la "forme" du problème.
- Le coeur de l'algorithme est une version itérative du FFT binaire de Cooley-Tukey.
- O(n log n), même si la dimension est un nombre premier.

Référence: http://www.fftw.org

Multiplication entière

Multiplication entière. Étant donnés deux entiers de n bits $a = a_{n-1} \ldots a_1 a_0$ et $b = b_{n-1} \ldots b_1 b_0$, calculer leur produit $c = a \times b$.

Algorithme (convolution).

- On utilise deux polynômes $A(x)$ et $B(x)$: $a = A(2)$, $b = B(2)$.
- On calcule $C(x) = A(x) \times B(x)$.
- On évalue $C(2) = a \times b$.
- Temps d'exécution: $O(n \log n)$ opérations arithmétiques complexes.

$$A(x) = a_0 + a_1 x + a_2 x^2 + L + a_{n-1} x^{n-1}$$

$$B(x) = b_0 + b_1 x + b_2 x^2 + L + b_{n-1} x^{n-1}$$

Théorème. [Schönhage-Strassen 1971] $O(n \log n \log \log n)$ opérations binaires.

En pratique. [GNU Multiple Precision Arithmetic Library] GMP prétend être "the fastest bignum library on the planet." Elle utilise un algorithme brutal, Karatsuba, ou FFT, selon la dimension du problème.

Un petit extra

Décomposition de la matrice de Fourier

Décomposition de la matrice de Fourier.

$$F_n = \begin{bmatrix} 1 & 1 & 1 & 1 & L & 1 \\ 1 & w^1 & w^2 & w^3 & L & w^{n-1} \\ 1 & w^2 & w^4 & w^6 & L & w^{2(n-1)} \\ 1 & w^3 & w^6 & w^9 & L & w^{3(n-1)} \\ M & M & M & M & O & M \\ 1 & w^{n-1} & w^{2(n-1)} & w^{3(n-1)} & L & w^{(n-1)(n-1)} \end{bmatrix}$$

$$P_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$D_4 = \begin{bmatrix} w^0 & 0 & 0 & 0 \\ 0 & w^1 & 0 & 0 \\ 0 & 0 & w^2 & 0 \\ 0 & 0 & 0 & w^3 \end{bmatrix}$$

$$F_n = \begin{bmatrix} I_{n/2} & D_{n/2} \\ I_{n/2} & -D_{n/2} \end{bmatrix} \begin{bmatrix} F_{n/2} & 0 \\ 0 & F_{n/2} \end{bmatrix} P_n^T$$

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$y = F_n a = \begin{bmatrix} I_{n/2} & D_{n/2} \\ I_{n/2} & -D_{n/2} \end{bmatrix} \begin{bmatrix} F_{n/2} a_{even} \\ F_{n/2} a_{odd} \end{bmatrix}$$