FFT [Cormen p827-841]

Rq on peut mettre des indices aux ω plutôt que des exposants

Schéma de la FFT

/ Lagrange, la donnée d'1 pol de deg n est $\leftrightarrow$ à la donnée de $(n+1)$ couples (point, valeur). Selon la représentation (coefficients ou (points, valeurs)), la complexité de la multiplication n'est pas la même :

$O(n^2)$ pour version coefficients

$O(n)$ pour version (point, valeur) via $(x_k, P(x_k)) \times (x_k, Q(x_k)) = (x_k, P(x_k) \times Q(x_k))$

↑ Rq : il faut $2n+1$ évaluat° ici.

Mais on peut faire 1 multiplication avec version coeff + efficace en traduisant de la version (point valeur) :

$P = (p_k)_k$, $Q = (q_k)_k$ —évaluat° (en $O(n \ln(n))$)→ $(x_k, P(x_k))_k$ et $(x_k, Q(x_k))_k$

× en $O(n^2)$ ↓ ; × en $O(n)$ ↓

$PQ = \left(\sum_{i+j=k} p_i q_j\right)_k$ ← interpolation (en $O(n \ln(n))$) — $(x_k, PQ(x_k))_k$

Rq En règle générale, l'évaluation de $A = \sum_{i=0}^{n} a_i X^i$ en x_0 se fait via Horner ($P(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \dots (+ x_0 a_n)))$) en temps $O(n)$. Pour évaluer A en $2n+1$ points, il faut donc 1 temps $O(n^2) \Rightarrow$ pas d'amélioration /n multiplication naïve. On va être + futés en évaluant en des points bien choisis. = les racines n^e de 1

au pire on compète pas des zéros et $|\text{nouveaux entiers}| \leq 2|\text{anciens entiers}|$

(TH) Soit P un polynôme de degré $n-1$ tel que n est une puissance de 2

On note $\omega^0, \omega^1, \dots, \omega^{n-1}$ les racines n^e de l'unité ; $\omega = e^{\frac{2i\pi}{n}}$; $(p_0, \dots, p_{n-1})$ les coeff de P et $(y_0, \dots, y_{n-1})$ les valeurs $(P(\omega^0), \dots, P(\omega^{n-1}))$.

= les x_i bien choisis !

Il est possible de calculer la transformée de Fourier discrète de $(p_0, \dots, p_{n-1})$ = le vecteur $(y_0, \dots y_{n-1})$ en temps $O(n \ln(n))$ via l'algo de transformat° de Fourier rapide = FFT

démo On peut écrire $P = P_0(x^2) + x P_1(x^2)$ en séparant termes d'indice pair / impair

Évaluer P en les ω^i revient donc à évaluer P_0 et P_1 en $(\omega^0)^2, \dots (\omega^{n-1})^2$

On, $\in \{(\omega^0)^2, \dots (\omega^{n-1})^2\} = \{$racines $\frac{n}{2}$ è de 1$\}$

En effet, $\varphi \left\{ \begin{array}{ccc} U_n & \longrightarrow & U_{n/2} \\ x & \longmapsto & x^2 \end{array} \right.$ est surjective (elle est bien déf d'ailleurs) car, comme n est une puissance de 2, $n = 2^\ell$ donc si $\alpha = e^{\frac{2ik\pi}{2^{\ell-1}}} \in U_{\frac{n}{2}}$,

$\alpha = \varphi\left(\underbrace{e^{\frac{2ik\pi}{2^\ell}}}_{\in U_n} \right)$ donc on a $\#\mathrm{Im}(\varphi) = \#\{\omega^2 \mid \omega \in U_n\} = \frac{n}{2}$

Donc on revient à évaluer A_0 et P_1 (de deg $\frac{n}{2}$) en les racines $\frac{n}{2}$ è de 1 $\Rightarrow$

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$ ← encore à vérifier

$\Rightarrow$ évaluation en $O(n\ln(n))$ par le th fondamental

algo FFT($P = (P_0, \dots, P_{n-1})$) =

si n = longueur$(P) = 1$, renvoyer P

sinon : $\omega_n = e^{\frac{2i\pi}{n}} = e^{-\frac{2i\pi}{n}}$; $\omega = 1$; $P_0 = (a_0, a_2, \dots, a_{n-2})$ et $P_1 = (a_1, a_3, \dots, a_{n-1})$; = division

récursion : puis $y_0 = \mathrm{FFT}(P_0)$ et $y_1 = \mathrm{FFT}(P_1)$

fusion : pour $k = 0$ à $\left(\frac{n}{2} - 1\right)$:

$y_k = (y_0[k] + \omega y_1[k])/n$

$y_{k+\frac{n}{2}} = (y_0[k] - \omega y_1[k])/n$

$\omega = \omega\omega_n$

renvoyer $(y_k)_k$

en noir = les modifs pour faire l'interpolation

Explications pour la fusion : les prop de symétrie des racines de 1 permettent une petite amélioration de la fusion : comme $-\omega^k = e^{i\pi} e^{\frac{2ik\pi}{n}} = e^{\frac{2i\pi\left(k+\frac{n}{2}\right)}{n}} = \omega_{k+\frac{n}{2}}$,

on peut, grâce à $P_0(\omega^k)$ et $P_1(\omega^k)$ calculer $y_k = P_0(\omega^k) + \omega^k P_1(\omega^k)$

et $y_{k+\frac{n}{2}} = P(-\omega^k) = P_0\left((-\omega^k)^2\right) - \omega^k P_1\left((-\omega^k)^2\right)$

$= P_0(\omega^k) - \omega^k P_1(\omega^k)$

On a bien un coût pour fusion + division en $O(n)$.

(prop) L'interpolation se fait de la même manière

démo / déf

$$\begin{pmatrix} y_0 \\ \vdots \\ y_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & & \omega^{n-1} \\ \vdots & (\omega^1)^2 & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 1 & (\omega^1)^{n-1} & & & (\omega^{n-1})^{n-1} \end{pmatrix}}_{= V_n = \text{matrice de Vandermonde donc inversible.}} \begin{pmatrix} P_0 \\ \vdots \\ P_{n-1} \end{pmatrix}$$

Connaissant la transformée de Fourier discrète de $PQ = (y_0, \dots y_{2n-1})$

on peut donc retrouver les coeff $(a_0, \dots a_{n-2})$ via $(a_i)_i = V_n^{-1}(y_i)_i$

⚠ notations malheureuses

Or, on a $V_n = \left(\omega^{kj}\right)_{0 \le k,j \le n-1}$. On mq $V_n^{-1} = \left(\frac{\omega^{-kj}}{n}\right)_{0 \le k,j \le n-1}$

En effet, $V_n V_n^{-1} = I_n$ car

le coeff (k,j) de cette matrice égale $\sum_{l=0}^{n-1} \omega^{kl} \frac{\omega^{-lj}}{n}$

$$= \sum_{l=0}^{n-1} \frac{1}{n} \times \omega^{l(k-j)} = \begin{cases} 1 \text{ si } k = j \\ \frac{1}{n} \times \frac{1-(\omega^{k-j})^n}{1-\omega^{k-j}} = 0 \text{ si } k \neq j \end{cases}$$

(série géométrique + $\omega^n = 1$)

on en déduit que $a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega^{-jk}$ = évaluation de $\sum_{k=0}^{n-1} y_k X^k$ en ω^{-j}

puis division par n

Donc en faisant les modifs en moi de l'algo FFT, on peut retrouver les coeffs de PQ

D'où :

algo pour multiplication rapide de polynômes :

Mult (P, Q) =

compléter P et Q avec des coeff = 0 pour avoir $n-1 = \deg(P) = \deg(Q)$ et n est une puissance de 2

Évaluer P et Q en les $(\omega^k)_k$ où ω = racine $(2n-1)$ ème de l'unité

Évaluer $\left(\frac{P(\omega^k)\,Q(\omega^k)}{n}\right)_k$ en les $(\omega^{-k})_k$.

(représentation coeffs)

(ça va marcher en complétant P et Q avec un tas de 0 pour qu'ils soient de "degré" $2n-1$ et $2n$ est bien une puissance de 2)

(comme P et Q sont de vrai deg $\le n-2$, deg $(PQ) \le 2n-2$ donc avec $2n$ évaluations ça suffit pour retrouver PQ.)

de complexité $O(n \ln(n))$