Transformée de Fourier rapide

Références : Cormen, Leiserson, Rivest, Stein, *Algorithmique*, p 827-841

• Problème :
Soient A et B deux polynômes, on veut les multiplier. On suppose qu'ils sont tous deux de degré inférieur ou égal à $n-1$ avec n une puissance de 2.
Si $A(X) = \sum_{j=0}^{n-1} a_j X^j$ et $B(X) = \sum_{j=0}^{n-1} b_j X^j$, alors

$$C(X) = A(X)B(X) = \sum_{j=0}^{2n-1} c_j X^j \text{ avec } c_j = \sum_{k=0}^{j} a_k b_{j-k}.$$

Si on calcule le produit ainsi, on fait $\mathcal{O}(n^2)$ opérations.
On voudrait néanmoins calculer ce produit en $\mathcal{O}(n\log(n))$ opérations.

• Multiplier avec Force, Fougue et Tendresse.
Le polynôme C est de degré inférieur à $2n-1$, il est donc entièrement déterminé par les valeurs prises sur une base du dual de $\mathbb{C}_{2n-1}[X]$.
Si on se donne $2n$ complexes x_i distincts alors les formes linéaires $L_i : P \mapsto P(x_i)$ sont linéairement indépendantes et forment donc une base du dual.
On a ainsi un isomorphisme

$$\begin{array}{rcl} \mathbb{C}_{2n-1}[X] & \to & \mathbb{C}^{2n} \\ P & \mapsto & (P(x_1), ..., P(x_{2n})) \end{array}.$$

De ce constat, on obtient une nouvelle manière de multiplier :

1. On évalue A et B sur une certaine famille $(x_i)_i$ à $2n$ éléments.
2. On multiplie les vecteurs obtenus pour obtenir $C(x_i) = A(x_i)B(x_i)$ ($\mathcal{O}(n)$ opérations).
3. On interpole C à partir des valeurs $C(x_i)$.

Pour évaluer un polynôme en une valeur, on peut utiliser la méthode de Horner :

$$A(x_0) = a_0 + x_0(a_1 + x_0(a_2 + ...)).$$

On a ainsi une évaluation en $\mathcal{O}(n)$ étapes, et donc l'évaluation en $2n$ points de A et B nous coûte $\mathcal{O}(n^2)$ opérations.
On ne parle même pas de l'interpolation qui nécessite l'inversion d'une matrice ($\mathcal{O}(n^3)$) ou le calcul direct par les polynômes de Lagrange ($\mathcal{O}(n^2)$).
La solution à notre problème est donnée par la FFT.

• Évaluation par FFT :
L'idée est de choisir pour les (x_i) les racines de l'unité. On pose $\omega_k^j = \exp\left(\frac{2ik\pi}{j}\right)$. Montrons que l'on peut évaluer A et B en les ω_{2n}^j - $j \in [0, 2n-1]$ - en $\mathcal{O}(n\log(n))$ opérations.

Faisons le pour A :
On pose $A^{[0]}(X) = \sum a_{2j} X^j$ et $A^{[1]}(X) = \sum a_{2j+1} X^j$. Alors $A(X) = A^{[0]}(X^2) + XA^{[1]}(X^2)$.
On doit évaluer $A^{[0]}$ et $A^{[1]}$ en les $(\omega_{2n}^j)^2$. Or $(\omega_{2n}^j)^2 = \omega_n^j = (\omega_{2n}^{n+j})^2$, donc il ne nous reste qu'à faire $2n$ évaluations sur des polynômes de degrés inférieur à $n/2$.

Voici l'algorithme $FFT(A)$:

1. n=degré de A
2. si $n = 1$, retourner A
3. Définition de $A^{[0]}$ et $A^{[1]}$
4. $y^{[0]} = FFT(A^{[0]})$ et $y^{[1]} = FFT(A^{[1]})$
5. Pour $j = 0 : n-1$ faire
6. $\quad y(j) = y^{[0]}(j) + \omega_{2n}^j y^{[1]}(j)$

7. $y(j+n) = y^{[0]}(j) - \omega_{2n}^j y^{[1]}(j)$
8. renvoyer y

L'avant dernière étape vient juste du fait que

$$A(\omega_{2n}^{n+j}) = A^{[0]}(\omega_n^j) + \omega_{2n}^{n+j} A^{[1]}(\omega_n^j),$$

et $\omega_{2n}^{n+j} = \exp(i\pi)\omega_{2n}^j = -\omega_{2n}^j$.

• Complexité :
On note T la complexité de l'algorithme FFT. Alors, comme les évaluations de $y^{[0]}$ et $y^{[1]}$ requièrent $\mathcal{O}(n)$ opérations, on a

$$T(n) = 2T\left(\frac{n}{2}\right) + \mathcal{O}(n).$$

Écrivons $n = 2^k$, alors on a en divisant par 2^k :

$$\frac{T(2^k)}{2^k} = \frac{T(2^{k-1})}{2^{k-1}} + \mathcal{O}(1).$$

On en déduit que $\dfrac{T(2^k)}{2^k} = \mathcal{O}(k)$, donc $T(2^k) = \mathcal{O}(k2^k)$.
On a juste à écrire $k = \log_2(n)$ pour obtenir

$$T(n) = \mathcal{O}(n\log(n)).$$

• Interpolation :
Il nous reste à présent à interpoler C à partir des $C(\omega_{2n}^j)$ en $\mathcal{O}(n\log(n))$ opérations.
Cela revient à résoudre le système linéaire suivant :

$$\begin{pmatrix} C(\omega_{2n}^0) \\ C(\omega_{2n}^1) \\ \vdots \\ C(\omega_{2n}^{2n-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega_{2n}^1 & \cdots & \omega_{2n}^{2n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \omega_{2n}^{2n-1} & \cdots & \omega_{2n}^{(2n-1)(2n-1)} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{2n-1} \end{pmatrix}.$$

Appelons V la matrice de Vandermonde à inverser.
On remarque que V^{-1} est donnée par $V_{i,j}^{-1} = \dfrac{\omega_{2n}^{-ij}}{2n}$.
En effet,

$$(V^{-1}V)_{i,j} = \sum_{k=0}^{2n-1} \frac{\omega_{2n}^{-ik}}{2n}\omega_{2n}^{kj} = \frac{1}{2n}\sum_{k=0}^{2n-1}\omega_{2n}^{k(j-i)} = \delta_{i,j}.$$ [1]

Finalement, on a

$$c_j = \frac{1}{2n}\sum_{k=0}^{2n-1} C(\omega_{2n}^k)\omega_{2n}^{-kj}.$$

Pour calculer C, on doit donc évaluer le polynôme $\widetilde{C}(X) = \dfrac{1}{2n}\displaystyle\sum_{k=0}^{2n-1} C(\omega_{2n}^k)X^k$ en les $(\omega_{2n}^{-j})_j$. Mais cela, on sait le faire avec la FFT en $\mathcal{O}(n\log(n))$ opérations.

• Conclusion :
On a réussi à multiplier deux polynômes en $\mathcal{O}(n\log(n))$ opérations. Pour cela, on évalue A et B avec la FFT, on fait une multiplication coordonnée par coordonnée, puis on interpole $C = AB$ à nouveau avec la FFT.

Remarques : • Il existe beaucoup d'améliorations et de variantes de cet algorithme. On peut en trouver certaines dans le Cormen.
• La grande idée de ce développement est la suivante : "La transformée de Fourier transforme un produit de convolution en un produit.". Une fois cela acquis, tout est assez logique.

Adapté du travail de Alexandre Bailleul et Paul Alphonse.

1. C'est une série géométrique. Il suffit de calculer.