

TD 4 : Transformée de Fourier rapide

1 Opérations sur les polynômes

Question 1 : On considère deux ensembles A et B , contenant chacun n entiers compris entre 0 et $10n$. On souhaite calculer la somme cartésienne C de A et B , définie par

$$C = \{x + y \mid x \in A \wedge y \in B\} .$$

On veut trouver les éléments de C et le nombre de fois que chaque élément de C est obtenu comme somme d'éléments de A et B . Montrer qu'on peut résoudre ce problème en temps $O(n \log n)$.

Question 2 : Dédire une représentation par valeurs de $\tilde{A}(x) = \sum_{j=0}^{n-1} a_{n-1-j}x^j$ à partir d'une représentation par valeurs de $A(x) = \sum_{j=0}^{n-1} a_jx^j$, en supposant qu'aucun des points n'est 0.

Question 3 : Étant donnée une liste de valeurs z_0, z_1, \dots, z_{n-1} (avec répétitions possibles), montrer comment trouver les coefficients d'un polynôme P de degré borné par n qui s'annule uniquement sur les points donnés. La procédure trouvée devra s'exécuter en temps $O(n \log^2 n)$.

Question 4 : On suppose disposer d'un algorithme efficace calculant la division euclidienne de deux polynômes : plus précisément, étant donné P et Q de degrés inférieurs à $n - 1$, on sait calculer le reste de la division euclidienne de P par Q , noté $P \bmod Q$, en temps $O(n \log n)$. En remarquant que $P(z) = P \bmod (x - z)$, montrer que l'on peut calculer $(P(z_0), \dots, P(z_{n-1}))$ en temps $O(n \log^2 n)$.

2 FFT itérative, FFT parallèle

On cherche à donner une version itérative de l'algorithme récursif FFT (pour l'image des racines de l'unité par un polynôme) vu en cours.

Question 5 : Dessiner l'arbre des appels récursifs de l'algorithme FFT pour un polynôme de degré 7, $P = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$.

Question 6 : Écrire un algorithme RÉORDONNE qui prend en entrée une liste de n coefficients et renvoie cette liste triée en fonction de l'ordre d'apparition aux feuilles de l'arbre des appels récursifs de FFT. Par exemple, pour les coefficients $P = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$, le résultat doit être $(a_0, a_4, a_2, a_6, a_1, a_5, a_3, a_7)$. L'algorithme devra s'exécuter en temps $O(n)$.

Question 7 : En déduire un algorithme itératif FFT-ITÉRATIVE. Montrer qu'il a la même complexité que l'algorithme récursif.

Question 8 : Montrer qu'en utilisant $n/2$ processeurs en parallèle, le calcul peut se faire en temps $O(\log n)$ (une fois que la liste des coefficients a été réordonnée).

3 Calcul des n premières dérivées d'un polynôme en un point

Étant donnée la représentation par coefficients $(a_0, a_1, \dots, a_{n-1})$ d'un polynôme A et un point x_0 , on souhaite déterminer $A^{(k)}(x_0)$, la dérivée k -ième de A en x_0 , pour tous les $k \in \{0, 1, \dots, n-1\}$.

Question 9 : Connaissant des coefficients b_0, b_1, \dots, b_{n-1} tels que

$$A(x) = \sum_{j=0}^{n-1} b_j (x - x_0)^j ,$$

montrer comment calculer $A^{(k)}(x_0)$ pour tous les $k \in \{0, 1, \dots, n-1\}$ en temps $O(n)$.

Question 10 : Expliquer comment trouver les b_i de l'équation ci-dessus en temps $O(n \log n)$, connaissant $A(x_0 + \omega^k)$ pour $k \in \{0, 1, \dots, n-1\}$ et ω racine primitive n -ème de l'unité.

Question 11 : Démontrer que

$$A(x_0 + \omega^k) = \sum_{r=0}^{n-1} \left(\frac{\omega^{kr}}{r!} \sum_{j=r}^{n-1} f(j)g(j-r) \right) ,$$

où $f(j) = a_j j!$ et $g(l) = x_0^l / (l!)$.

Question 12 : Expliquer comment évaluer $A(x_0 + \omega^k)$ pour $k \in \{0, 1, \dots, n-1\}$ en temps $O(n \log n)$. (*Indication :* on pourra appliquer plusieurs FFT et essayer d'exprimer $A(x_0 + \omega^k)$ à l'aide d'un produit de polynômes.)

Question 13 : Que peut-on conclure ?

4 Application de FFT : filtrage

Question 14 : (DM) Un dispositif physique tel qu'un microphone, un oscilloscope, etc., est utilisé pour acquérir une suite de réels x_0, \dots, x_n assez longue. Lorsque le dispositif n'est pas de très bonne qualité et que les échantillons contiennent du "bruit", une mesure rudimentaire pour lutter contre ce phénomène consiste à appliquer un lissage gaussien, c'est-à-dire à remplacer chaque échantillon par une moyenne pondérée de ses voisins :

$$y_i = \frac{1}{Z} \sum_{j=-k}^k x_{i+j} \cdot e^{-j^2}$$

où k est un paramètre de largeur et Z un facteur de normalisation choisi convenablement. On remarque qu'il y a un problème pour les valeurs limites, mais il est résolu en ne calculant ni les k premières valeurs, ni les k dernières. Montrer comment calculer la suite y_k, \dots, y_{n-k+1} en temps $O(n \log n)$.