TD 4 : Transformée de Fourier rapide

Exercice 1 Déduire une représentation par valeurs de $A^{mir}(x) = \sum_{j=0}^{n-1} a_{n-1-j}x^j$ à partir d'une représentation par valeurs de $A(x) = \sum_{j=0}^{n-1} a_j x^j$, en supposant qu'aucun des points n'est 0.

Exercice 2 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

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

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 $O(n \log n)$.

Exercice 3 Étant donnée une liste de valeurs $z_0, z_1, \ldots, z_{n-1}$ – 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 $O(n \log^2 n)$.

FFT itérative, FFT parallèle

On cherche à donner une version itérative de l'algorithme récursif vu en cours.

Exercice 4 Dessiner l'arbre des appels récursifs de l'algorithme de la FFT pour un polynôme de degré 7 $P = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$.

Exercice 5 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. L'algorithme devra s'exécuter en temps $O(n)$.

Exercice 6 En déduire un algorithme itératif FFT-Itérative. Montrer qu'il a la même complexité que l'algorithme récursif.

Exercice 7 Montrer qu'en utilisant $n/2$ processeurs en parallèle, le calcul peut se faire en temps $O(\log n)$.

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

Exercice 8 Étant donnés la représentation par coefficients $(a_0, a_1, \ldots, a_{n-1})$ d'un polynôme A et un point x_0, on souhaite déterminer $A^{(k)}(x_0)$, la dérivée $t$ième de A en x_0, pour tous les $k \in \{0, 1, \ldots, n-1\}$.

1. Connaissant des coefficients $b_0, b_1, \ldots, b_{n-1}$ 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 $k \in \{0, 1, \ldots, n-1\}$ en temps $O(n)$.

2. Expliquer comment trouver les b_i de l'équation ci-dessus en temps $O(n \log n)$, connaissant $A(x_0 + \omega_n^k)$ pour $k \in \{0, 1, \ldots, n-1\}$.

3. Démontrer que

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

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

4. Expliquer comment évaluer $A(x_0 + \omega_n^k)$ pour $k \in \{0, 1, \ldots, n-1\}$ en temps $O(n \log n)$. (*Indication:* on pourra appliquer plusieurs FFT)

5. Conclure.