Université Bordeaux 1 — **Algèbre et calcul formel – Master**
Mathématiques — Année 2018–2019

FEUILLE D'EXERCICES n° 4

Exercice 1 – [FFT : DEUX EXEMPLES]
Nous considérons ici des polynômes de $K[X]$ où K est un corps. Soit n une puissance de 2 différente de 1 : $n = 2^k$ avec $k > 0$. On suppose que K possède une racine primitive n-ième de l'unité, qu'on note ω. . On appelle *transformée de Fourier discrète* d'un polynôme $R \in K[X]$ de degré $< n$, identifié au n-uplet $(R_0, \ldots, R_{n-1})$, le n-uplet

$$F_\omega(R) = (R(1), R(\omega), \ldots, R(\omega^{n-1})).$$

1) Dans $K = \mathbb{C}$, on choisit i comme racine quatrième de l'unité. Soit $P = x^3 + 2x^2 + 3x + 4 \in \mathbb{C}[x]$. Exécuter l'algorithme FFT vu en cours pour calculer $F_i(P)$.

2) Prenons maintenant $K = \mathbb{F}_{29}$.

a) Quelles sont les racines primitives quatrièmes de l'unité dans $\mathbb{F}_{29}$? Soit ω l'une de ces racines.

b) Soit $P = x^3 + 2x^2 + 3x - 1$ dans $\mathbb{F}_{29}$. Exécuter l'algorithme FFT vu en cours pour calculer $F_\omega(P)$.

3) Quelle est la plus petite extension de $\mathbb{F}_3$ contenant les racines primitives 16-ème de 1 ?

Exercice 2 – [DIVISER POUR RÉGNER : TOOM-COOK]

Soit R un polynôme de degré inférieur ou égal à 4. Si l'on connaît 5 valeurs de R, alors on sait reconstituer R par interpolation. Ici, on suppose que l'on connaît les valeurs de R en 0, 1, -1 et 2, et aussi le coefficients de X^4 dans R. Les deux premières questions montrent comment on peut reconstituer R par un calcul matriciel.

1) On note $R = a_4X^4 + a_3X^3 + a_2X^2 + a_1X + a_0$. Pour marquer l'analogie avec l'interpolation, on note aussi $R(\infty) = a_4$. Soient

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 16 & 8 & 4 & 2 & 1 \end{pmatrix} \quad \text{et} \quad M' = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -2 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{6} & \frac{1}{6} \\ -1 & -1 & \frac{1}{2} & \frac{1}{2} & 0 \\ 2 & -\frac{1}{2} & 1 & -\frac{1}{3} & -\frac{1}{6} \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Vérifier que $M' = M^{-1}$ et montrer les égalités

$$M\begin{pmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} R(\infty) \\ R(0) \\ R(1) \\ R(-1) \\ R(2) \end{pmatrix} \quad \text{et} \quad M'\begin{pmatrix} R(\infty) \\ R(0) \\ R(1) \\ R(-1) \\ R(2) \end{pmatrix} = \begin{pmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix}$$

2) Soient $P = 3X^2 + X - 1$, $Q = X^2 - 2X + 1$ et $R = PQ$. Calculer $R(\infty)$, $R(0)$, $R(1)$, $R(-1)$ et $R(2)$, puis reconstituer R en utilisant la question précédente.

Il aurait été plus simple de calculer R en multipliant P et Q de façon classique, mais un traitement récursif de cette méthode donne un algorithme plus efficace quand les degrés de P et de Q sont grands. C'est ce que nous étudions dans les questions qui suivent.

3) On suppose que P et Q sont des polynômes de $\mathbb{C}[X]$ de degrés strictement inférieur à $n = 3^r$. On note

$$P = \sum_{i=0}^{n-1} a_i X^i \quad \text{et} \quad Q = \sum_{i=0}^{n-1} b_i X^i.$$

On note $\mathit{Interp}(z_1, \ldots, z_5, Y)$ le polynôme R en la variable Y de degré inférieur ou égal à 4 dont les valeurs en $y_1 = \infty$, $y_2 = 0$, $y_3 = 1$, $y_4 = -1$ et $y_5 = 2$ sont $(z_1, \ldots, z_5)$.

Expliquer ce que fait l'algorithme *Toom-Cook* suivant.
Toom-Cook(P, Q, n) :

(1) Si $n = 1$, retourner PQ.

(2) Écrire P et Q sous la forme

$$P = (X^{n/3})^2 P_2 + X^{n/3} P_1 + P_0 \quad \text{et} \quad Q = (X^{n/3})^2 Q_2 + X^{n/3} Q_1 + Q_0,$$

où les P_i et les Q_j sont des polynômes de degrés inférieurs strictement à $n/3$.

(3) Soient les polynômes de $\mathbb{C}[X][Y]$

$$A = Y^2 P_2 + Y P_1 + P_0 \quad \text{et} \quad B = Y^2 Q_2 + Y Q_1 + Q_0.$$

(4) Pour i de 1 à 5, faire $z_i =$*Toom-Cook*$(A(y_i), B(y_i), n/3)$ (les z_i sont alors des polynômes en X).

(5) $\mathit{Produit} =$*Interp*$(z_1, z_2, z_3, z_4, z_5, Y)$.

(6) Retourner $R = \mathit{Produit}(X^{n/3})$

4) Quelle est la complexité algébrique de l'algorithme Toom-Cook ?

5) Revenons à l'algorithme de Karatsuba. Expliquer en quoi cet algorithme suit lui aussi une stratégie « évaluation-produit-interpolation ».

Exercice 3 – [DIVISER POUR RÉGNER : UNE GÉNÉRALISATION DU LEMME]
On rappelle le lemme qui permet parfois de calculer plus facilement la complexité si l'on utilise une approche du type "diviser pour régner".

Lemme. *Soit* $f : \mathbb{R}^+ \to \mathbb{R}^+$ *telle que*
— f *est bornée sur* $[0,1]$ *;*
— *quel que soit* $x \geqslant 1$*, on a* $f(x) \leqslant af(x/b) + Mx$ *où* a*,* $M > 0$ *et* $b > 1$*.*
Alors sur $[1,\infty[$ *on a*

$$f(x) = \begin{cases} O(x^{\ln a/\ln b}) & si \quad a > b \\ O(x \ln x) & si \quad a = b \\ O(x) & si \quad a < b \end{cases}$$

À l'aide d'un changement de variable approprié, établir la généralisation suivante.

Lemme bis. *Soit* $f : \mathbb{R}^+ \to \mathbb{R}^+$ *telle que*
— f *est bornée sur* $[0,1]$ *;*
— *quel que soit* $x \geqslant 1$*, on a* $f(x) \leqslant af(x/b) + Mx^r$ *où* a*,* M*,* $r > 0$ *et* $b > 1$*.*
Alors sur $[1,\infty[$ *on a*

$$f(x) = \begin{cases} O(x^{\ln a/\ln b}) & si \quad a > b^r \\ O(x^r \ln x) & si \quad a = b^r \\ O(x^r) & si \quad a < b^r \end{cases}$$

Exercice 4 – [DIVISER POUR RÉGNER, UN AUTRE EXEMPLE : STRASSEN]
Soient A et B deux matrices $n \times n$. On cherche à calculer $C = AB$ de façon économique. On suppose pour commencer (questions 1 et 2) que n est une puissance de 2. On divise A, B et C en 4 matrices $n/2 \times n/2$.

$$A = \begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}, \quad B = \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} \quad \text{et} \quad C = \begin{pmatrix} C_1 & C_2 \\ C_3 & C_4 \end{pmatrix}.$$

On pose

$$\begin{cases} P_1 &= A_1(B_2 - B_4) \\ P_2 &= (A_1 + A_2)B_4 \\ P_3 &= (A_3 + A_4)B_1 \\ P_4 &= A_4(B_3 - B_1) \\ P_5 &= (A_1 + A_4)(B_1 + B_4) \\ P_6 &= (A_2 - A_4)(B_3 + B_4) \\ P_7 &= (A_1 - A_3)(B_1 + B_2) \end{cases}$$

1) Exprimer C_2 à l'aide de P_1 et P_2, C_3 à l'aide de P_3 et P_4, C_1 à l'aide de $P_4 + P_5$ et $P_2 - P_6$ et C_4 à l'aide de $P_1 + P_5$ et $P_3 + P_7$.

2) Exprimer la relation de récurrence que vérifie la complexité algébrique c_n de l'algorithme et en déduire que

$$c_n = O\Big(n^{\ln 7/\ln 2}\Big).$$

3) Généraliser à n quelconque.

4) Comparer avec la complexité de l'algorithme naïf issu de l'application directe de la formule

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

Exercice 5 – [Racines primitives n-èmes de l'unité dans un anneau]
Soit A un anneau commutatif unitaire. Soit $n > 1$ un entier naturel. On dit que $\omega \in A$ est une racine primitive n-ème de l'unité si $\omega^n = 1$ et si pour tout diviseur premier l de n, l'élément $\omega^{n/l} - 1$ n'est pas un diviseur de 0. On suppose qu'il existe un tel élément ω dans A.

1) Soit k un entier tel que $1 < k < n$ et soit $d = \mathrm{pgcd}(k, n)$. Vérifier qu'il existe un diviseur premier l de n tel que d divise n/l.

2) Montrer qu'il existe un élément a de A tel que

$$a(\omega^d - 1) = \omega^{n/l} - 1.$$

En déduire que $\omega^d - 1$ n'est pas un diviseur de 0.

3) Montrer que $\omega^k - 1$ divise $\omega^d - 1$. En déduire que $\omega^k - 1$ n'est pas un diviseur de 0.

4) Montrer que

$$\sum_{j=0}^{n-1} \omega^{kj} = 0.$$