Université Bordeaux 1 — **Algèbre et calcul formel – Master**
Mathématiques — Année 2012–2013

FEUILLE D'EXERCICES n° 5

Travail sur machine

Exercice 1 – [KARATSUBA]
Rappelons le principe. On désire calculer le produit de deux polynômes P, $Q \in R[X]$ de degré(s) $< n$, où R est un anneau commutatif. L'approche naïve a une complexité algébrique en $O(n^2)$. Une façon d'améliorer ce résultat est la suivante. Considérons nos polynômes comme des polynômes de degré(s) $< 2^s$ où s est le plus petit entier tel que $n \leqslant 2^s$, i.e. $s = \lceil \log n / \log 2 \rceil$. Supposons $s > 0$ et écrivons

$$P = X^{2^{s-1}} P_1 + P_2 \quad \text{and} \quad Q = X^{2^{s-1}} Q_1 + Q_2,$$

où P_1, P_2, Q_1 et Q_2 sont des polynômes de degré(s) $< 2^{s-1}$. On a alors

$$\begin{aligned} PQ &= X^{2^s} P_1 Q_1 + X^{2^{s-1}} (P_1 Q_2 + P_2 Q_1) + P_2 Q_2 \\ &= X^{2^s} P_1 Q_1 + X^{2^{s-1}} \left((P_1 + P_2)(Q_1 + Q_2) - P_1 Q_1 - P_2 Q_2 \right) + P_2 Q_2, \end{aligned}$$

de telle sorte que nous avons juste à calculer trois produits

$$A = P_1 Q_1,\ B = P_2 Q_2 \text{ et } C = (P_1 + P_2)(Q_1 + Q_2)$$

de polynômes de degré(s) $< 2^{s-1}$. On utilise alors cette idée de façon récursive, ce qui conduit à un algorithme dont la complexité algébrique est en $O(n^{\log 3 / \log 2})$.

Pour simplifier, soit n une puissance de 2. On considère deux polynômes P et Q de degrés $< n$.

1) Écrire une procédure récursive `Karatsuba`(P, Q, n) utilisant le principe rappelé ci-dessus et renvoyant une liste correspondant à PQ.

2) Tester cette procédure sur des polynômes symboliques de degré 3.

3) La tester numériquement avec de gros polynômes que l'on fabriquera à l'aide de la procédure définie dans l'exercice 3 de la séance 3.

Exercice 2 – [FFT]
Soit n une puissance de 2 différente de 1 : $n = 2^k$ avec $k > 0$. Soit ω une racine primitive n-ième de l'unité, par exemple $\omega = e^{2i\pi/n}$. On rappelle que si R est un polynôme de $\mathbb{C}[X]$ de degré $< n$, que l'on identifiera au n-uplet $(R_0, \ldots, R_{n-1})$ on a

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

et que pour évaluer $DFT_\omega(R)$ on peut se ramener au calcul de deux DFT de degrés $< m = n/2$ par le biais des formules

$$\begin{cases} R(\omega^p) & = & \sum_{j=0}^{m-1} R_{2j}\alpha^{jp} + \omega^p \sum_{j=0}^{m-1} R_{2j+1}\alpha^{jp} \\ R(\omega^{p+m}) & = & \sum_{j=0}^{m-1} R_{2j}\alpha^{jp} - \omega^p \sum_{j=0}^{m-1} R_{2j+1}\alpha^{jp}. \end{cases}$$

où $0 \leqslant p < m$ et où $\alpha = \omega^2$.

Rédiger l'algorithme récursif s'appuyant sur cette remarque. La procédure dite `FFT` recevra en entrées R, ω et n, et retournera $DFT_\omega(R)$. On prendra garde à ne pas calculer ω^p à chaque étape de la boucle sur p. Pour cela, on pourra par exemple les stocker en amont.

Exercice 3 – [Produit rapide de polynômes par FFT]
Ici encore $n = 2^k$ avec $k > 0$. Soient P et Q deux polynômes de $\mathbb{C}[X]$ vérifiant $\deg(PQ) < n$. On identifiera encore P et Q aux n-uplets $(P_0, \ldots, P_{n-1})$ et $(Q_0, \ldots, Q_{n-1})$. On rappelle que l'on a alors

$$DFT_\omega(PQ) = DFT_\omega(P) \cdot DFT_\omega(Q)^1,$$

et que pour tout polynôme $R \in \mathbb{C}[X]$ de degré $< n$ on a

$$DFT_{\omega^{-1}}(DFT_\omega(R)) = DFT_\omega(DFT_{\omega^{-1}}(R)) = nR.$$

Écrire une procédure prenant en arguments P, Q et n et retournant PQ, procédure qui prendra bien sûr appui sur la procédure `FFT` de l'exercice 1.

Remarque. Pour s'assurer que $\deg(PQ) < n$ on pourra imposer à P et Q d'être tous deux de degrés $< m = n/2$.

[1] ici $(u_i)_{0\leqslant i<n} \cdot (v_i)_{0\leqslant i<n} = (u_i v_i)_{0\leqslant i<n}$.