Université de Bordeaux — **Algèbre et calcul formel – Agrégation**
Mathématiques — 2017–2018

FEUILLE D'EXERCICES n° 4

Multiplication de polynômes : Karatsuba, FFT

On propose ici d'écrire des procédures de multiplication de polynômes. Chacune de ces procédures prendra comme entrée les deux polynômes sous forme de listes et rendra le produit également sous forme de liste. Ces procédures seront faciles à appliquer à des polynômes grâce aux transformations liste-polynôme et polynôme-liste.

Exercice 1 – [Karatsuba]
On désire calculer le produit de deux polynômes $P, Q \in R[X]$ de degrés $< n$, où R est un anneau commutatif et n une puissance de 2. L'approche naïve a une complexité algébrique de $O(n^2)$ opérations dans R. Une façon d'améliorer ce résultat est la suivante. Si $n = 1$, P et Q sont des polynômes constants et on les multiplie normalement. Sinon, on pose $m = n/2$. On écrit

$$P = X^m P_1 + P_0 \quad \text{and} \quad Q = X^m Q_1 + Q_0,$$

où P_1, P_0, Q_1 et Q_0 sont des polynômes de degrésstrictement inférieurs à m. On a alors

$$\begin{aligned} PQ &= X^n P_1 Q_1 + X^m (P_1 Q_0 + P_0 Q_1) + P_0 Q_0 \\ &= X^n P_1 Q_1 + X^m \left((P_1 + P_0)(Q_1 + Q_0) - P_1 Q_1 - P_0 Q_0\right) + P_0 Q_0, \end{aligned}$$

de telle sorte que nous avons juste à calculer trois produits

$$P_0 Q_0, \quad P_1 Q_1 \quad \text{et} \quad (P_0 + P_1)(Q_0 + Q_1)$$

de polynômes de degrés $< m$. On utilise cette idée de façon récursive, ce qui conduit à un algorithme dont la complexité algébrique est en $O(n^{\log_2 3})$.

1) Soit donc n une puissance de 2. Écrire une procédure récursive `Karatsuba(`P, Q, n`)` utilisant le principe rappelé ci-dessus et renvoyant le polynôme PQ.

Dans cette procédure, P et Q seront rentrés comme des listes, et le polynôme PQ obtenu sera également une liste.

2) Tester cette procédure sur les polynômes $a_0 + a_1 x$ et $a_2 + a_3 x$ de $\mathbb{Z}[a_0, a_1, a_2, a_3][x]$.

3) Écrire ensuite une procédure `MulK(A,P,Q)` qui, étant donnés deux polynômes P et Q définis dans l'anneau de polynômes `A` utilise `Karatsuba` pour donner le produit PQ, lui aussi défini dans `A`.

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

5) La tester numériquement avec de gros polynômes pris au hasard. On rappelle que si `A` est un objet défini comme un ensemble éventuellement muni d'une structure, on peut obtenir un élément au hasard dans A en utilisant `A.random_element`).

Exercice 2 – [FFT] Soit $n = 2^k$ une puissance de 2, avec $k > 0$. Soit ω une racine primitive n-ième de l'unité, par exemple $\omega = e^{2i\pi/n}$. On définit un isomorphisme de $\mathbb{C}$-algèbres $\mathcal{F}_\omega : \mathbb{C}[X]/(X^n - 1) \to \mathbb{C}^n$ par

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

On identifiera une classe $\bar{R} \in \mathbb{C}[X]/(X^n - 1)$ avec son représentant $R \in \mathbb{C}[X]$ de degré $< n$, ainsi qu'avec le n-uplet $(R_0, \ldots, R_{n-1})$ de ses coefficients, dans $\mathbb{C}^n$.

On évalue $\mathcal{F}_\omega(R)$ en calculant récursivement deux $\mathcal{F}$ de degrés $< m = n/2$ par le biais des formules

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

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

Programmer l'algorithme récursif s'appuyant sur cette remarque. La procédure `FFT` recevra en entrées R, ω et n, et retournera $\mathcal{F}_\omega(R)$. Là encore, le polynôme R sera défini par une liste. On prendra garde à ne pas calculer ω^p à chaque étape de la boucle sur p.

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 $(Q_0, \ldots, Q_{n-1})$ formés de leurs coefficients. On rappelle que l'on a alors

$$\mathcal{F}_\omega(PQ) = \mathcal{F}_\omega(P) \cdot \mathcal{F}_\omega(Q)^1,$$

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

$$\mathcal{F}_{\omega^{-1}}(\mathcal{F}_\omega(R)) = \mathcal{F}_\omega(\mathcal{F}_{\omega^{-1}}(R)) = nR.$$

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 précédent.

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

Exercice 4 – [FFT sur un corps fini]
Chercher à appliquer l'algorithme de l'exercice 2 à un corps $\mathbb{F}_p$, par exemple $\mathbb{F}_{257}$ et $n = 16$. Essayer aussi l'exercice 3 pour de tels exemples.

Exercice 5 – [La fonction `fft` de sage]

1) Expérimenter les commandes suivantes.

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

```
A=[RR(1) for i in range(8)]
s=IndexedSequence(A,range(8))
t=s.fft();t
lt=t.list();lt
```

2) Comparer les résultats donnés par cette fonction `fft` avec les résultats de l'exercice 2.