Université Bordeaux 1 — **Algèbre et calcul formel – Agrégation**
Mathématiques — Année 2021–2022

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 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.

1) Montrer que

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

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

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

2) 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 – [La fonction `fft` de sage]

1) Expérimenter les commandes suivantes.

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

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

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

Exercice 5 – [FFT SUR UN CORPS FINI]

1) Chercher à appliquer l'algorithme de l'exercice 2 à des corps $\mathbb{F}_p$ et $\mathbb{F}_q$, par exemple $\mathbb{F}_{257}$ puis $\mathbb{F}_{81}$ avec $n = 16$. Essayer aussi l'exercice 3 pour de tels exemples.

2) Même exercice sur $\mathbb{F}_{81}$, toujours avec $n = 16$.

Exercice 6 – [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.

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

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

Alors $M' = M^{-1}$. De plus,

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

1) 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 (à la machine) 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.

2) 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) $Produit =$*Interp*$(z_1, z_2, z_3, z_4, z_5, Y)$.

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

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

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

Exercice 7 – [Filtres]
Soit f une fonction périodique de période 1 de $\mathbb{R}$ dans $\mathbb{C}$. Il suffit de connaître f sur $[0, 1]$ pour connaître f. Soient N une puissance de 2 et $w = e^{2i\pi/N}$. Soit

$$F = \left[f(0), f\left(\frac{1}{N}\right), \ldots, f\left(\frac{N-1}{N}\right) \right].$$

Pour tout n,

$$F[n] = \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{F}_{w^{-1}}(F)[k] w^{nk}.$$

Plus simplement, si l'on note

$$\hat{F} = \frac{1}{N} \mathcal{F}_{w^{-1}}(F),$$

$$F[n] = \sum_{k=0}^{N-1} \hat{F}[k] w^{nk}.$$

Ainsi, la transformation $F \mapsto \hat{F}$ permet de passer de F à la suite des composantes fréquencielles de F.

1) Étudions un exemple simple. Tous les calculs seront faits en flottant. Soit

$$f(t) = \sin(18\pi t) + 3\cos(10\pi t).$$

Soient $N = 16$ et $w = e^{i\pi/8}$. On définit F comme ci-dessus.

a) En utilisant votre fonction pour la fft, calculer $\hat{F}$.

b) Représenter sur un graphique la ligne brisée définie par les points

$$\left[(0, F[0]), \ldots, \left(\frac{15}{16}, F[15]\right)\right]$$

(on pourra utiliser la fonction `line`).

Représenter sur un autre graphique les points

$$\left[(0, |\hat{F}[0]|), \ldots, \left(\frac{15}{16}, |\hat{F}[15]|\right)\right]$$

en utilisant la fonction `point`. Commenter ce graphique.

c) Calculer $\mathcal{F}_w(\hat{F})$, et représenter sur un graphique la ligne brisée définie par cette fonction. La comparer avec la ligne brisée correspondant à F.

Pour mieux les comparer, on peut les placer sur le même graphique en s'inspirant du modèle suivant, où A et B sont des listes, et où x est la liste des $\dfrac{i}{16}$.

```
GrapheA=line([(x[i],A[i]) for i in range(16)],color='red')
GrapheB=line([(x[i],B[i]) for i in range(16)])
GrapheA+GrapheB
```

Les deux premières lignes sont des affectations. La troisième trace les deux lignes sur un même graphique. Cela peut aussi s'écrire

```
GrapheA=line(zip(x,A),color='red')
GrapheB=line(zip(x,B))
GrapheA+GrapheB
```

d) Introduisons un bruit dans le signal F.

```
def h():
    return ZZ.random_element(-500,500)/1000
```

définit une fonction aléatoire. Soit le bruit

```
B=[h() for i in range(16)]
```

Ajoutons ce bruit à F.

```
FB=[F(i)+B[i] for i in range(16)]
```

Faire un graphe dessinant la ligne définie par FB et la comparer à celle de F. On suppose qu'au lieu du signal F, on ait reçu le signal brouillé FB. On va essayer de reconstituer F, en considérant que les fréquences ajoutée par le bruit sont de faible amplitude.

e) Calculer $\widehat{FB}$ et dessiner l'ensemble des points correspondants. Comparer avec les points donnés par $\hat{F}$.

f) Soit G définie par : $G[i] = \widehat{FB}[i]$ si $|\widehat{FB}[i]| > 1/4$ et $G[i] = 0$ sinon. Calculer $\mathcal{F}_w(G)$ et comparer avec F. Commenter.

2) Refaire l'exercice avec $N = 1024$ et $w = e^{2i\pi/1024}$.

3) Soit f la fonction périodique de période 1 définie sur $[0, 1[$ par

$$f(t) = 1920(t - 1/6)(t^2 - 1/2)(t - 1/2)(t - 7/8)(t - 1/10)(t - 1)t.$$

Refaire l'exercice précédent sur cette fonction avec $N = 1024$ et $w = e^{2i\pi/1024}$. Quand on nettoie le signal brouillé (question 1.f), on peut modifier le seuil à partir duquel on met $\widehat{FB}[k]$ à 0.

4) On peut aussi choisir de filtrer les fréquences situées dans certains intervalles. Reprendre la question 3 en mettant à 0 les $\widehat{FB}[k]$ tels que $k \in [[11, 1013]]$.

5) Expérimenter la fonction définie sur $[0, 1[$ par

$$f(t) = 1920(t - 1/6)(t^2 - 1/2)(t - 1/2)(t - 7/8)(t - 1/10)(t - 1).$$