FEUILLE D'EXERCICES 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 – [Gérer l'aléa]
Commençons par quelques commandes utiles pour gérer l'aléa dans Sage.

1) La plupart des ensembles de Sage ont une méthode `.random_element()`, qui permet de tirer des éléments aléatoires dans l'ensemble. Si l'ensemble est fini, l'élément est tiré uniformément dans l'ensemble. Sinon il est tiré selon une distribution choisie par Sage (vous pouvez obtenir plus d'info grâce à l'aide, en faisant `ZZ.random_element?` par exemple). Testez par exemple

```
ZZ.random_element()
ZZ.random_element(a,b) (l'élément tiré sera uniforme dans [a,b[)
QQ['x'].random_element()
QQ['x'].random_element(d) (renvoie un polynôme aléatoire de degré d)
Integers(7).random_element()
GF(32).random_element()
```

2) Lorsqu'on veut faire une démo et que certains bouts du code utilisent de l'aléa, on aime bien fixer l'aléa, pour que la démo fasse toujours la même chose. C'est faisable en sage en utilisant `set_random_seed(42)` (vous pouvez remplacer 42 par votre entier préféré). Attention, si vous évaluez plusieurs fois la même cellule, et que vous avez mis votre `set_random_seed(42)` dans une cellule précédente, l'aléa va quand même changer. Par exemple

```
set_random_seed(42)
print(ZZ.random_element())
print(ZZ.random_element())
```

Vous devriez obtenir -31, puis 5. Normalement, n'importe qui utilisant Sage avec la même version que vous devrait obtenir les mêmes valeurs.

Exercice 2 – [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

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

où P_1, P_0, Q_1 et Q_0 sont des polynômes de degrés strictement 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.

1) Quelle est la complexité de cet algorithme ? (en terme d'opérations sur le corps de base)

2) 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.

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

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

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

6) La tester numériquement avec de gros polynômes pris au hasard.

Exercice 3 – [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 $\alpha = \omega^2$.

1) 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.

2) Quelle est la complexité de votre algorithme ? (En terme d'opérations dans le corps de base K).

Exercice 4 – [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

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

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 $< m = n/2$.

1) Quelle est la complexité de cet algorithme de multiplication ? (En terme d'opérations dans $\mathbb{C}$)

Exercice 5 – [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 3.

Exercice 6 – [FFT SUR UN CORPS FINI]
Appliquer l'algorithme de l'exercice 3 à un corps $\mathbb{F}_p$, par exemple $\mathbb{F}_{257}$ et $n = 16$. Essayer aussi l'exercice 4 pour de tels exemples.

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 flottants. 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 $\frac{i}{16}$.

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

Les deux premières lignes sont des affectations. La troisième trace les deux lignes sur un même graphique. On pourrait aussi utiliser le fait que Sage affiche par défaut la dernière ligne de la cellule et ne pas mettre le `plot` dans la dernière ligne. Par contre, si vous rajoutez une commandes après dans la cellule, votre graphique ne s'affichera plus... Cela peut aussi s'écrire

```
GrapheA = line(zip(x,A),color='red')
GrapheB = line(zip(x,B))
plot(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 $\left|\widehat{FB}[i]\right| > 1/4$ 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 $k \in [[11, 1013]]$ (cela revient à filtrer les hautes fréquences, responsables des oscillations serrées).

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

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

Remarque. Ça ne marche pas bien sur cette fonction, avez-vous une explication ?
(Indice : regardez les valeurs de $f(0)$ et $f(1)$)