Devoir maison

CPES2: Algorithmique et Applications

Pierre SENELLART

À rendre pour le 21 novembre 2021

Ce devoir maison est formé d'un unique problème noté sur 20 points. Vous devez d'ici la date limite (21 novembre 2021, 23h59) soumettre votre devoir sur le site Moodle du cours, à l'endroit indiqué, sous la forme d'un unique PDF (qui peut contenir un document produit électroniquement ou des scans ou photos de documents papier manuscrits, tant que le résultat est lisible et prend moins de 50 MB). Tout retard dans le rendu sera pénalisé de points en moins.

Multiplication de polynômes

On note $\mathbb{N}[X]$ l'ensemble des polynômes à coefficients entiers. Tout polynôme P de $\mathbb{N}[X]$ non nul peut s'écrire de manière unique sous la forme $P(X) = \sum_{i=0}^{d} c_i X^i$ avec c_i entier et $c_d \neq 0$. Le *degré* de P est d. Par convention, le polynôme nul P_0 se note $P_0(X) = 0$ et son degré est de -1. Dans tout le problème, on considérera que les nombres entiers manipulés sont de taille constante et donc que les opérations arithmétiques sur les nombres entiers se font en $O(1)$.

L'objet de ce problème est d'étudier différentes méthodes pour calculer le résultat de la multiplication de deux polynômes de $\mathbb{N}[X]$.

1. (2 points) On souhaite représenter un polynôme par une classe Python `Polynome` $\mathbb{N}[X]$ comportant deux propriétés :

   `_degre`, le degré du polyôme ;

   `_coeff`, le tableau des coefficients représenté sous forme de liste Python ;

   Écrire le constructeur d'une telle classe, qui prend en argument un tableau de coefficients. Ainsi, `Polynome([1,0,1,1])` créera un objet représentant le polynôme $X^3 + X^2 + 1$.

2. (1 point) Ajouter à cette classe une méthode magique `__getitem__` permettant d'accéder au i-ème coefficient du polynôme P en écrivant `P[i]`.

3. (2 points) Ajouter à cette classe une méthode magique `__call__` permettant d'utiliser un objet de classe polynôme comme une fonction afin d'évaluer un polynôme en un point. Ainsi, `P(42)` doit retourner la valeur du polynôme P en le point 42. On pourra utiliser la méthode de Horner pour réaliser cette évaluation (voir TP 2). Quelle est la complexité de cette méthode ?

4. (3 points) Ajouter à cette classe une méthode magique `__str__` permettant d'afficher le polynôme sous une forme lisible en listant les monômes par ordre décroissant (p. ex., « `X^3 + X^2 + 1` » pour le polynôme $X^3 + X^2 + 1$ ou bien « `X^5 - X^4 - X^3 + 2X^2 - 2X + 1` » pour le polynôme $X^5 - X^4 - X^3 + 2X^2 - 2X + 1$).

5. (2 points) Ajouter à cette classe une méthode magique `__add__` permettant de calculer le polynôme résultant de l'addition des deux polynômes en entrée. Quelle est la complexité de cette méthode ? Grâce à cette méthode, le code suivant :

```
P = Polynome([1,-2,1])
Q = Polynome([1,0,1,1])
R = P+Q
print(R)
```

doit produire l'affichage « `X^3 + 2X^2 - 2X + 2` ».

6. (1 point) Ajouter à cette classe une méthode magique `__sub__` permettant de calculer le polynôme résultant de la soustraction des deux polynômes en entrée. Quelle est la complexité de cette méthode ? Grâce à cette méthode, le code suivant :

```
P = Polynome([1,-2,1])
Q = Polynome([1,0,1,1])
R = P-Q
print(R)
```

doit produire l'affichage « `- X^3 - 2X` ».

7. (2 points) Ajouter à cette classe une méthode magique `__mul__` permettant de calculer le polynôme résultant de l'addition des deux polynômes en entrée, *le plus simplement possible, sans chercher à optimiser ce calcul* ; on appellera cette méthode l'algorithme **N**. Quelle est la complexité de cette méthode ? Grâce à cette méthode, le code suivant :

```
P = Polynome([1,-2,1])
Q = Polynome([1,0,1,1])
S = P*Q
print(S)
```

doit produire l'affichage « `X^5 - X^4 - X^3 + 2X^2 - 2X + 1` ».

8. (2 points) On considère l'algorithme suivant (appelé algorithme **B**) pour calculer la multiplication de deux polynômes P et Q de $\mathbb{N}[X]$ dont le degré maximal est d :
   - Si $d = 0$, on multiplie P et Q comme deux nombres entiers.
   - Sinon, soit $m = \lceil \frac{d}{2} \rceil$ (où $\lceil \cdot \rceil$ dénote la fonction *partie entière supérieure*) ; on écrit P et Q comme, chacun, la somme de deux polynômes :

$$\begin{cases} P = P_2 X^m + P_1 \\ Q = Q_2 X^m + Q_1 \end{cases}$$

     avec P_1, P_2, Q_1, Q_2 des polynômes de degré $< m$. On pose alors :

$$\begin{cases} R_1 = P_1 \times Q_1 \\ R_{12} = P_2 \times Q_1 + P_1 \times Q_2 \\ R_2 = P_2 \times Q_2 \end{cases}$$

     et on calcule $P \times Q$ comme $R_2 X^{2m} + R_{12} X^m + R_1$ en appliquant l'algorithme récursivement.

   Calculer la complexité asymptotique de l'algorithme **B** et la comparer à celle de l'algorithme **N**. Comme d'habitude on pourra faire l'hypothèse pour simplifier l'analyse que les parties entières tombent toujours juste.

9. (2 points) En développant $(P_1 + P_2) \times (Q_1 + Q_2)$, proposer une autre manière de calculer R_{12} (avec les notations de la question précédente). En déduire un autre algorithme (que l'on appellera algorithme **K**) pour le calcul du produit de deux polynômes. Calculer la complexité asymptotique de l'algorithme **K** et la comparer à celle des algorithmes **N** et **B**.

10. (3 points) Ajouter à la classe `Polynome` une méthode statique `algok` permettant d'utiliser l'algorithme **K** pour calculer le produit de deux polynômes passés en argument. (On utilisera donc cette méthode avec `Polynome.algok(P,Q)`.)

    En utilisant une extension des techniques vues à la fin du TP 6, il est en fait possible d'avoir une complexité bien meilleure à celle des techniques vues dans ce problème : plus précisément, la multiplication de deux polynômes de degré n peut se faire en $O(n \log n)$. Pour plus de détails, voir le chapitre 30 du livre de référence.