Arithmétique des polynômes

Étienne Miquey
etienne.miquey@ens-lyon.fr

Dans ce TP, nous allons nous intéresser à la multiplication de polynômes, qui en fait fonctionne sur les mêmes idées que celle des des grands entiers. Pour permettre aux polynômes d'avoir des degrés arbitraires, nous les représenterons par des listes, qui contiendront les coefficients du polynôme et que nous ordonnerons par degrés croissants [1]. La première partie est là pour vous faire manipuler cette représentation et construire une petite bibliothèque standard, tout en ayant conscience de la complexité des opérations. La seconde partie vous présente un algorithme plus que classique (donc à connaître) de multiplication. Enfin, les parties 3 et 4 (plus longue) sont indépendantes, faites ce que vous pouvez/voulez en fonction de votre temps.

Jusqu'à la partie 4, le type de vos coefficients importe peu, vous êtes donc complétement libres de prendre ce que vous voulez, et notamment des entiers. Cependant, si vous voulez éviter d'avoir à tout réadapter à ce moment là, je vous conseille de travailler avec la bibliothèque `num`. Cela vous pemettra d'avoir vos coefficients dans le corps des rationnels et d'éviter quelques petites mauvaises surprises algébriques.

1 Une petite bibliothèque standard

Chacune des questions suivantes ne devrait vous prendre que peu de temps. Vous avez le droit de vous chronométrer pour voir. Essayez d'avoir le souci d'une représentation unique et normalisée de vos polynômes.

Question 1. Écrire les polynômes $X^2 + 4X + 3$ et $X^3 - 2X^2 + X$ dans cette représentation, ainsi que des fonctions renvoyant le degré d'un polynôme, son coefficient dominant et tout ce qui peut vous sembler utile dans le genre.

Question 2. Écrire des fonctions d'addition et soustraction pour les polynômes. Quelle est la complexité $A(n)$ d'une telle fonction ?

Question 3. Écrire une fonction de multiplication qui prendra en argument un polynôme P, un élément a et un entier n et qui retournera la multiplication de $P(X)$ par le monôme aX^n. Complexité ?

Question 4. Écrire l'algorithme de multiplication naïf de deux polynômes. Quelle en est la complexité $M(n)$?

Question 5. Écrire une fonction de dérivation formelle d'un polynôme. Complexité ?

Question 6. Enfin, pour terminer les outils de base, écrire une fonction qui réalisera l'évaluation d'un polynôme en un point. Je rappelle à tout hasard que celle-ci doit se faire d'après la méthode de Hörner [2]. Complexité ?

2 Karatsuba

Afin de diminuer la complexité de la multiplication de polynômes, nous allons nous servir d'une technique de programmation très classique : *diviser pour régner*. Pour cela, on va décomposer les polynômes à multiplier P et Q de degrés strictement inférieurs à $2n$ en $P = P_1 + P_2 \cdot X^n$ et $Q = Q_1 + Q_2 \cdot X^n$ avec P_1, P_2, Q_1 et Q_2 de degrés strictement inférieurs à n.

Question 7. Écrire une formule qui réduit $P \cdot Q$ en multiplications des polynômes P_1 , P_2 , Q_1 et Q_2 de tailles inférieurs à n.

Question 8. Programmer un algorithme récursif de multiplication utilisant la formule précédente. Comparez sa complexité avec $M(n)$.

On peut raffiner cette méthode grâce à la remarque suivante de Karatsuba : le terme intermédiaire de $P \cdot Q$ s'écrit $P_1 \cdot Q_2 + P_2 \cdot Q_1 = (P_1 + P_2) \cdot (Q_1 + Q_2) - P_1 \cdot Q_1 - P_2 \cdot Q_2$. On échange donc deux multiplications et une addition contre une multiplication et quatre additions.

Question 9. En vous servant de la remarque précédente, programmer un algorithme récursif de multiplication à la Karatsuba.

Question 10. Écrire la formule de récurrence qui définit la complexité de cet algorithme. Sa solution est-elle meilleure que celles trouvées jusqu'alors ?

1. *Si vous ne comprenez toujours pas pourquoi après la question 3, posez-vous la question !*
2. *Basée sur la remarque que* $a_nX^n + a_{n-1}X^{n-1} + \ldots + a_0 = X \cdot (\cdots (X \cdot (a_nX + a_{n-1}) + a_{n-2}) \ldots) + a_0$

3 Kronecker

La substitution de Kronecker associe à un polynôme un entier qui le représente. Cela part du simple constat que, si les coefficients d'un polynôme $P(X) = \sum_{i=0}^{n} a_i X^i$ sont tous positifs et bornés par un entier q (autrement dit on travaille dans un $\mathbb{Z}/p\mathbb{Z}$), alors la donnée de $P(q)$ suffit à retrouver les a_i en effectuant des divisions euclidiennes successives ($a_0 = P(q) \mod q$, etc), comme pour une écriture de nombres dans une base.

Question 11. La fonction d'encodage en suivant cette idée n'est autre que celle d'évaluation codée précédemment. Écrire la fonction de décodage correspondante.

Question 12. Si l'on considère q assez grand, en fait on peut même effectuer la multiplication de polynômes via leur représentation de Kronecker. Écrire de ce fait une fonction de multiplication en servant de cela. Que dire de sa complexité ?

4 Évaluation et interpolation

Nous allons avoir pour cette section besoin de travailler avec des coefficients dans un corps commutatif, afin de pouvoir définir la division euclidienne. Pour cela, vous pouvez soit définir les rationnels à votre sauce, soit vous resservir de la librairie `num` comme lors du précédent TP.

Question 13. Écrire un algorithme de division de polynômes basé sur l'algorithme d'Euclide. Complexité ?

4.1 Évaluation multi-points

Étant donné $P(x)$ un polynôme de degré $n-1$ et $(x_0, \ldots, x_{n-1})$ des points deux à deux distincts, l'objectif de l'évaluation multi-points est de calculer le vecteur $(P(x_0), \ldots, P(x_{n-1}))$.

Question 14. Quelle serait la complexité d'un algorithme naïf ?

Afin d'obtenir un algorithme efficace, nous allons à nouveau utiliser une technique de diviser pour régner. Supposons P de degré $2p-1$. On pose

$$M_{k,l}(X) = \prod_{i=k}^{l} (X - x_i)$$

et on considère la division de P par $M_0 = M_{0,p-1}$ et $M_1 = M_{p,2p-1}$: $P = Q_0 \cdot M_0 + R_0$ et $P = Q_1 \cdot M_1 + R_1$

Question 15. Que peut-on dire des $P(x_i)$?

Question 16. En déduire un algorithme d'évaluation rapide et l'implémenter.

En fait, on peut encore améliorer la chose en précalculant l'arbre des sous-produits $M_{i,j}$. J'imagine que vous devriez savoir à cet époque de l'année ce qu'est un arbre et comment le représenter en machine, mais sinon un truc du genre

```
type arbre = Noeud of polynom * arbre * arbre | Feuille of polynome;;
```

devrait faire l'affaire.

Question 17. Écrivez une fonction qui, étant donnée la liste des $(x_0, \ldots, x_{n-1})$, précalcule l'arbre des sous-produits. Modifiez ensuite l'évaluation rapide en conséquence.

À ce stade là, on peut montrer (avec l'aide de gros théorèmes pas beaux mais bien utiles) que la complexité de l'évalution multi-points précédente est un $O(M(n)log(n))$, ce qui est donc bien mieux.

4.2 Interpolation

Soient $(x_0, \ldots, x_{n-1})$ des nombres deux à deux distincts. Soient $(y_0, \ldots, y_{n-1})$ des nombres quelconques. On cherche à calculer le polynôme interpolateur des y_i aux points x_i , c'est-à-dire $P(X)$ de degré $n-1$ tel que $P(x_i) = y_i$ pour tout i. Je vous rappelle que le polynôme interpolateur de Lagrange est usuellement donné par la formule :

$$P(X) = \sum_{i=0}^{n-1} y_i \frac{L_i(X)}{L_i(x_i)} \text{ où } L_i(X) = \prod_{j \neq i} (X - x_j)$$

Question 18. Quelle serait la complexité d'un algorithme d'interpolation basée sur cette formule ?

Ceci serait donc trop lourd à mettre en place. Nous allons donc nous intéresser à une méthode plus efficace.

En notant $M(X) = \prod_i (X - x_i)$, on peut réécrire la formule précédente ainsi :

$$P(X) = \sum_{i=0}^{n-1} c_i \frac{M(X)}{X - x_i}$$

Question 19. En se servant de M', que valent les c_i ? Déduisez de ceci un algorithme de calcul des c_i. Quelle est sa complexité ?

Question 20. Étant donnés des coefficients c_i , on cherche à évaluer rapidement la somme

$$\sum_{i=0}^{n-1} c_i \frac{M(X)}{X - x_i}$$

En utilisant une démarche récursive et les polynômes $M_{0,\frac{n}{2}-1}$ et $M_{\frac{n}{2},n-1}$ vus précédement, proposez et implémentez un algorithme rapide pour calculer cette somme. Normalement, vous devriez obtenir quelque chose en $O(M(n)log(n))$.

Question 21. En mettant le tout bout-à-bout, écrire une fonction d'interpolation par un polynôme. Comparez la complexité obtenue avec celle de la question 18.