

CMA : Devoir à la maison

Rendre les copies à votre enseignant de groupe avant Noël

Problème : Composition des polynômes

Soit $f = f(x)$, $h = h(t)$ deux polynômes de degré $\deg f, \deg h \leq n - 1$. L'objectif du problème est de calculer leur composition

$$F = F(t) = f(h(t)) \pmod{t^n},$$

ou, en d'autres termes, faire la substitution $x = h(t)$.

Le calcul mod t^n implique que dans chaque opération tous les degrés supérieurs ou égaux à n peuvent être éliminés (la question 7 fera une exception à cette règle : on y considérera des polynômes de degré supérieur à n). Aussi, les degrés des polynômes restent bornés par n . Nous ne répéterons plus dans chaque exercice qu'il s'agit de calcul mod t^n .

Nous utiliserons les notations suivantes :

$P(n)$ – la complexité de la multiplication des polynômes de degré $\leq n - 1$;

$M(n)$ – la complexité de la multiplication des matrices de taille $n \times n$.

Outre n , nous utiliserons aussi deux paramètres m et k tels que $mk = n$.

Algorithme A : schéma de Horner. Soit

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}.$$

Calculer $F = f(h)$ selon la formule suivante :

$$F(t) = a_0 + h(t)(a_1 + h(t)(a_2 + h(t)(a_3 + \dots + h(t)(a_{n-2} + h(t)a_{n-1}) \dots))).$$

Question 1. Quelle est la complexité $t_A(n)$ de l'Algorithme A ? Donner la réponse en utilisant la fonction $P(n)$.

Question 2. Montrer que le polynôme f peut s'écrire

$$f(x) = \sum_{i=0}^{k-1} g_i(x)x^{mi},$$

où chaque polynôme g_i est de degré $\deg g_i \leq m - 1$.

Soit A une matrice de taille $k \times m$ dont les lignes représentent les coefficients des polynômes g_i , et soit B une matrice de taille $m \times n$ dont les lignes représentent les coefficients des polynômes $1, h, h^2, \dots, h^{m-1}$.

Question 3. (a) Montrer que la matrice AB , de taille $k \times n$, contient dans la i -ème ligne ($i = 0, \dots, k-1$) les coefficients du polynôme $r_i(t) = g_i(h(t))$.

(b) Montrer que

$$F(t) = \sum_{i=0}^{k-1} r_i(t) \cdot h(t)^{mi}.$$

Le résultat de la question 3 permet d'utiliser l'algorithme suivant :

Algorithme B.

1. En supposant que n soit un carré, choisir $m = k = \sqrt{n}$.
2. Calculer h^2, h^3, \dots, h^m .
3. Calculer la matrice AB , en faisant m multiplications des matrices carrées de taille $m \times m$.
4. Calculer

$$F(t) = f(h(t)) = \sum_{i=0}^{k-1} r_i(t) \cdot (h(t)^m)^i$$

en utilisant le schéma de Horner.

Question 4. (a) Exprimer la complexité $t_B(n)$ de l'Algorithme B en fonction de n , en utilisant les fonctions $P(n)$ et $M(n)$.

(b) On admet que $P(n) = O(n \log n)$ (la multiplication des polynômes en utilisant la transformée de Fourier rapide), et que $M(n) = O(n^\alpha)$. Quelle est alors la complexité $t_B(n)$ de l'Algorithme B ? Concrétiser la réponse si $\alpha = 3$ (la multiplication "classique"), $\alpha = 2.807$ (la méthode de Strassen), $\alpha = 2.376$ (la méthode "record" de Coppersmith–Winograd).

Soit $\deg f \leq p-1$, $\deg h \leq q-1$. Si un des deux degrés est nettement inférieur à n , l'algorithme suivant, de type "diviser pour régner", peut donner un avantage par rapport à l'algorithme précédent. On suppose que p est une puissance de 2.

Algorithme C.

1. Calculer $h^2, h^4, h^8, \dots, h^{p/2}$.

2. Couper le polynôme f en deux blocs :

$$f(x) = f_0(x) + x^{p/2} \cdot f_1(x), \quad \deg f_0, \deg f_1 \leq p/2 - 1.$$

3. Calculer

$$f(h(t)) = f_0(h(t)) + h(t)^{p/2} \cdot f_1(h(t))$$

en appliquant récursivement la même méthode aux aux polynômes f_0 et f_1 .

Question 5. (a) Pour mesurer la complexité t_C de l'Algorithme C, prendre comme paramètre $N = pq$ et écrire une récurrence pour $t_C(N)$. Montrer que

$$t_C(N) = O(P(N) \log N).$$

(b) Montrer que si $P(n) = n \log n$ et $N = an$, $a \leq n$, alors

$$t_C(N) = O(aP(n) \log n).$$

Soit $h = h_0 + h_1$ où $\deg h_0 = m-1$ et $h_1 = t^m \cdot p(t)$ avec $\deg p \leq n-m-1$. Selon la formule de Taylor,

$$f(h) = f(h_0) + f'(h_0)h_1 + \frac{f''(h_0)}{2!}h_1^2 + \dots$$

Question 6. Montrer que, quand on fait le calcul mod t^n , il suffit, pour obtenir une égalité dans la formule de Taylor, de prendre les termes du développement jusqu'au $(k-1)$ -ème, c'est-à-dire, terminer le développement par

$$\frac{f^{(k-1)}(h_0)}{(k-1)!}h_1^{k-1}.$$

(Rappel : $k = n/m$.)

Les dérivées successives $f^{(i)}(h_0)$ peuvent être calculées en utilisant la formule de dérivation des fonctions composées :

$$(f^{(i)}(h_0))' = f^{(i+1)}(h_0) \cdot h_0'.$$

Mais, à cause du fait que la dérivation abaisse les degrés, il nous faudra, pour calculer la k -ème dérivée de $f(h_0)$ jusqu'au degré $n-1$, connaître plus de n termes des dérivées précédentes.

Admettons que la division (avec le reste) des polynômes de degré $\leq n-1$ peut se faire en temps $O(P(n))$.

Question 7. Montrer que si on connaît

$$f^{(i)}(h_0(t)) \pmod{t^{n+k-i}},$$

et si $h_0'(0) \neq 0$, alors le calcul de

$$f^{(i+1)}(h_0(t)) \pmod{t^{n+k-(i+1)}}$$

prend le temps $O(P(n+k-i)) = O(P(n))$ (expliquer la dernière égalité !).

Toutes ces observations mènent à l'algorithme suivant :

Algorithme D.

1. Calculer

$$\frac{h_1^2}{2!}, \frac{h_1^3}{3!}, \dots, \frac{h_1^{k-1}}{(k-1)!}.$$

2. Utiliser l'Algorithme C pour calculer

$$f(h_0(t)) \pmod{t^{n+k-1}}.$$

3. Pour $i = 1, 2, \dots, k-1$ calculer les dérivées

$$f^{(i)}(h_0(t)) \pmod{t^{n+k-1-i}}.$$

4. Calculer

$$f(h) = f(h_0) + f'(h_0)h_1 + \frac{f''(h_0)}{2!}h_1^2 + \dots + \frac{f^{(k-1)}(h_0)}{(k-1)!}h_1^{k-1}.$$

Question 8. (a) Montrer que la complexité de l'Algorithme D est

$$t_D(n) = O((k + m \log n)P(n)).$$

(b) Qu'est-ce que cela donne pour $P(n) = O(n \log n)$ et pour

$$m \approx \left(\frac{n}{\log n} \right)^{1/2} ?$$

FIN DE L'ÉNONCÉ