SESSION DE 1999

COMPOSITION DE MATHÉMATIQUES-INFORMATIQUE

(Epreuve commune aux ENS : Lyon et Cachan - Groupes M, I et MP)

Durée : 3 heures

L'usage de calculatrices électroniques de poche à alimentation autonome, non imprimantes et sans document d'accompagnement, est autorisé. Cependant, une seule calculatrice à la fois est admise sur la table ou le poste de travail, et aucun échange n'est autorisé entre les candidats.

Tournez la page S.V.P.

1 Introduction

Le problème propose de concevoir et d'analyser des algorithmes travaillant sur des nombres entiers relatifs (sauf indication contraire, "nombre entier" sera synonyme de "nombre entier relatif"). Les opérations autorisées pour manipuler des entiers sont les suivantes (a et b représentent des variables de l'algorithme):

- les trois opérations arithmétiques $a+b$, $a-b$, et $a \times b$.
- le calcul du quotient a div b et du reste a mod b de la division euclidienne de a par b;
- l'affectation, notée $a := b$;
- les tests d'égalité ou d'inégalité.

Etant donné un entier x, taille(x) est par définition[1] égal à $\log(2+|x|)$. taille(x) est donc une approximation de la taille de la représentation binaire de x; on remarquera en particulier que taille(x) ≥ 1 pour tout entier x. Les entiers manipulés étant de taille arbitrairement grande, il n'est pas réaliste de supposer que les opérations ci-dessus peuvent être effectuées en temps constant. On considérera dans ce problème que le coût d'une opération est égal à la somme de la taille de ses opérandes. Le coût d'un algorithme sur une entrée donnée est par définition égal à la somme du coût des opérations effectuées par cet algorithme. Dans certaines questions on s'intéresse aussi au nombre d'opérations effectuées par un algorithme, que nous appellerons complexité de l'algorithme. On prendra garde de ne pas confondre le coût d'un algorithme avec sa complexité.

Exemple: méthode de Hörner.
L'algorithme suivant permet d'évaluer un polynôme $f(X) = \sum_{i=0}^{d} a_{d-i} X^i$ en un point x.

$y = a_0$
pour i de 1 à d faire $y := y \times x + a_i$
retourner y.

Dans cet exemple f est donné par sa représentation dense, c'est à dire par la liste de tous ses coefficients $a_0, a_1, \ldots, a_d$. On notera taille-dense(f) $= \sum_{i=0}^{d}$ taille(a_i).

Le nombre d'opérations arithmétiques effectuées par la méthode de Hörner est polynomial en d (il est en fait exactement égal à $2d$). Comme les entiers calculés au cours de cet algorithme restent de taille polynomiale en taille-dense(f) + taille(x), on peut voir (nous invitons les candidats à s'en persuader) que le coût de l'algorithme est lui aussi polynomial en taille-dense(f) + taille(x). Ceci signifie qu'il existe une fonction polynomiale p tel que pour

[1]Dans ce problème, tous les logarithmes sont en base 2; comme d'habitude, $\mathbb{N}$, $\mathbb{Z}$ et $\mathbb{Q}$ désignent respectivement l'ensemble des entiers naturels, des entiers relatifs et des nombres rationnels.

toute entrée (f, x) de l'algorithme, son coût est majoré par $p(\text{taille-dense}(f) + \text{taille}(x))$.

Dans ce problème on s'intéresse surtout à la représentation creuse des polynômes, dans laquelle f est donné par la liste de ses coefficients *non nuls*. Plus précisément, f sera donné sous la forme:

$$f(X) = \sum_{i=1}^{k} a_i X^{\beta_i} \tag{1}$$

avec $\beta_1 > \beta_2 > \cdots > \beta_{k-1} > \beta_k \geq 0$ et tous les a_i non nuls (c'est-à-dire sous la forme d'une liste $(a_1, \beta_1), \ldots, (a_k, \beta_k)$ triée suivant les β_i). Par définition, $\text{taille-creuse}(f) = \sum_{i=1}^{k} (\text{taille}(a_i) + \text{taille}(\beta_i))$.

Les algorithmes demandés dans certaines questions seront décrits de manière simple et précise (comme par exemple la méthode de Hörner ci-dessus). Aucun programme écrit dans tel ou tel langage de programmation n'est par contre attendu. On rappelle enfin qu'une réponse, même juste, est sans valeur si elle n'est pas justifiée correctement.

2 Calcul de puissances

Dans cette partie on étudie des algorithmes qui permettent de calculer x^n en $O(\log n)$ opérations (avec x un entier et n un entier positif).

1. Supposons que les chiffres de l'écriture binaire de n sont disponibles dans un tableau $C[0..k]$, ordonné des poids faibles vers les poids forts ($C[0]$ est donc le chiffre des unités, et $C[k] = 1$).

   (a) Donnez une majoration et une minoration de k en fonction de n.

   (b) Donnez un algorithme de complexité $O(\log n)$ pour le calcul de x^n. L'algorithme devra parcourir le tableau C des poids forts vers les poids faibles.

2. Donnez un autre algorithme de calcul de x^n en complexité $O(\log n)$ parcourant C des poids faibles vers les poids forts.

   On introduira des variables y et z ainsi qu'un indice de boucle i tel qu'après l'itération i de la boucle les conditions suivantes sont vérifiées: $z = x^{2^{i+1}}$ et $y = x^{M(i)}$, avec $M(i) = \sum_{j=0}^{i} C[j]2^j$.

3. Réécrire cet algorithme en supposant que n est maintenant contenu dans une variable entière (qu'on pourra appeler également n par abus de langage). Comme dans le reste du problème, on ne manipulera les variables entières qu'avec les "opérations autorisées" décrites dans l'introduction.

4. (a) Combien de multiplications sont elles effectuées par chacun des trois algorithmes précédents pour calculer x^{15} ?

   (b) Soit $n \geq 1$ un entier. On appelle "chaîne de multiplications de longueur k pour le calcul de X^n" une suite $X^{a_0}, X^{a_1}, \ldots, X^{a_k}$ de $k+1$ monômes en l'indéterminée X vérifiant les propriétés suivantes:

Tournez la page S.V.P.

(i) $a_0 = 1$ et $a_k = n$.

(ii) Pout tout entier i tel que $1 \leq i \leq k$ il existe deux entiers u et v tels que $X^{a_i} = X^{a_u} \times X^{a_v}$ et $0 \leq u \leq v < i$.

Donnez les chaînes de multiplications pour le calcul de X^{15} correspondant aux trois algorithmes précédents. Existe-t-il une chaîne plus courte ?

5. Existe t-il un entier $n \geq 1$ et un algorithme qui pour tout entier x calcule x^n avec une complexité strictement inférieure à $\log n$?

3 Racines d'un polynôme creux

Dans cette partie on étudie des algorithmes de calcul des racines entières d'un polynôme f à coefficients entiers donné par sa représentation creuse (1). On supposera que f est de degré au moins 1. Les racines seront toujours comptées sans multiplicité.

On appelle *problème du signe* le problème suivant: étant donné un polynôme $f \in \mathbb{Z}[X]$ donné sous forme creuse et un entier x, déterminer le signe de $f(x)$. Dans cette partie on admettra que le problème du signe peut être résolu par un algorithme de coût polynomial en taille-creuse(f)+taille(x). Une démonstration de ce résultat est proposée dans la partie 4. On travaillera avec les conventions suivantes: pour tout entier a, signe(a) $= 1$ si $a > 0$, signe(a) $= 0$ si $a = 0$, et signe(a) $= -1$ si $a < 0$.

Soit $M > 0$ un entier et soit $\mathcal{C} = \{[u_i, v_i]\}_{1 \leq i \leq N}$ une liste d'intervalles à extrémités entières ($u_i, v_i \in \mathbb{Z}$) vérifiant les propriétés suivantes:

(i) Pour tout $i \leq N - 1$ on a $u_i < u_{i+1}$. Une liste d'intervalles vérifiant cette propriété est dite *ordonnée*.

(ii) Pour tout $i \leq N$ on a soit $v_i = u_i$, soit $v_i = u_i + 1$.

(iii) $u_1 \geq -M$ et $v_N \leq M$.

On dit que que $\mathcal{C}$ encadre les racines de f dans l'intervalle $[-M, M]$ si pour toute racine réelle ξ de f telle que $\xi \in [-M, M]$, il existe $i \leq N$ tel que $\xi \in [u_i, v_i]$.

1. Montrez que f a au plus $2k - 1$ racines réelles (on pourra raisonner par récurrence sur k).

2. On peut écrire f sous la forme $f(x) = x^\alpha p(x)$ avec $\alpha \geq 0$ un entier et p un polynôme tel que $p(0) \neq 0$. La dérivée p' de p est appelée *pseudo-dérivée* de f.

   Soit $\mathcal{C}' = \{[u'_i, v'_i]\}_{1 \leq i \leq N}$ une liste d'intervalles qui encadre les racines de p' dans l'intervalle $[-M, M]$. Montrez que pour chaque $i < N$, p a une racine dans l'intervalle $]v'_i, u'_{i+1}[$ si et seulement $p(v'_i)p(u'_{i+1}) < 0$.

3. En déduire un algorithme polynomial en taille-creuse(f) + taille(M) + N qui, étant donné f, M et $\mathcal{C}'$, construit une liste d'intervalles $\mathcal{C}$ encadrant

les racines de f dans $[-M, M]$. Pour décrire cet algorithme, les candidats pourront utiliser des phrases telles que: "J'insère l'intervalle I dans la liste ordonnée d'intervalles $\mathcal{C}$" sans avoir à expliquer comment une telle opération est effectuée.

4. Montrez que toutes les racines entières de f divisent a_k, sauf au plus l'une d'entre elles.

5. Soit $d_1, d_2, \ldots, d_k$ la suite des pseudo-dérivées successives de f, définie de la manière suivante: pour $1 \leq i \leq k-1$, d_{i+1} est la pseudo-dérivée de d_i, et $d_1 = f$.

   (a) Montrez que la suite des représentations creuses de $d_1, d_2, \ldots, d_k$ peut se calculer en complexité polynomiale en k.

   (b) Montrez que cette suite peut se calculer en coût polynomial en taille-creuse(f).

6. Montrez que les racines entières de f peuvent être calculées avec un coût polynomial en taille-creuse(f).

7. Dans cette question uniquement, on suppose f donné par sa représentation dense. Soit $x \in \mathbb{Q}$ une racine de f qui s'écrit sous la forme $x = p/q$ avec $p \in \mathbb{Z}$, $q \in \mathbb{N}^*$, p et q premiers entre eux. Que peut on dire de q ? En déduire un algorithme de coût polynomial en taille-dense(f) qui détermine les racines rationnelles de f.

4 Signe d'un polynôme creux

Dans cette partie on étudie le coût du problème du signe, défini au début de la partie 3.

1. Donnez un algorithme qui, étant donné un entier x et la représentation creuse d'un polynôme f, calcule $f(x)$ avec une complexité polynomiale en taille-creuse(f). On pourra s'inspirer de la méthode de Hörner décrite dans l'introduction.

2. On peut utiliser l'algorithme de la question précédente pour résoudre le problème du signe. Sur l'exemple de la famille de polynômes $f_n(x) = x^{2n} + x^n + 1$, minorez et majorez le coût de cette solution en fonction de n et de taille(x). Ce coût est-il polynomial en taille-creuse(f_n) + taille(x)?

3. Supposons donnés un polynôme $f \in \mathbb{Z}[X]$, un entier x et un entier naturel T tel que $|f(x)| < T$. Montrez que $f(x)$ peut être calculé par un algorithme de coût polynomial en taille-creuse(f) + taille(x) + taille(T). On pensera à utiliser l'opération *mod*.

4. Montrez qu'il existe un algorithme qui, étant donnés deux entiers $x \geq 1$ et $\alpha \geq 0$, calcule avec un coût polynomial en taille(x) + taille(α) un entier $l \geq 1$ tel que $2^{l-1} \leq x^\alpha \leq 2^{l+1}$.

Tournez la page S.V.P.

Indication: on admettra que pour tout entier $n \geq 1$, il est possible de calculer en coût polynomial en $n + \text{taille}(x)$ un entier y_n tel que:

$$|y_n/2^n - \log x| \leq 1/2^{n-1}.$$

5. Dans cette question on étudie un algorithme de coût polynomial en taille-creuse(f) + taille(x) pour le calcul du signe de f en un entier $x > 0$. On posera $s_i = \sum_{j=1}^{i} a_j x^{\beta_j - \beta_i}$ pour $i = 1, \ldots, k$.

   (a) Quel lien y a t-il entre le signe de $f(x)$ et le signe des s_i ?

   (b) Etablir une relation de récurrence entre s_i et s_{i-1}.

   (c) En déduire un algorithme de coût polynomial en taille-creuse(f) + taille(x) pour le calcul du signe de $f(x)$.

   Indication: on pourra calculer pour $i = 1, \ldots, k$ une suite de couples $(v_i, V_i) \in \mathbb{N}^2$ et une suite de signes $\sigma_i \in \{-1, 0, 1\}$ vérifiant les propriétés suivantes:

$$\begin{cases} s_i \in [2^{v_i}, 2^{V_i}] & \text{si } \sigma_i = 1 \\ s_i \in [-2^{V_i}, -2^{v_i}] & \text{si } \sigma_i = -1 \\ s_i = 0 & \text{si } \sigma_i = 0 \end{cases} \tag{2}$$

   et

$$0 \leq V_i - v_i \leq 3i - 2. \tag{3}$$

6. Comment peut-on évaluer le signe de f en un entier $x \in \mathbb{Z}$ arbitraire ?