ENS LYON INFORMATIQUE

EPREUVE D'INFORMATIQUE FONDAMENTALE ET DE MATHEMATIQUES APPLIQUEES

Durée 3 heures
(Note sur 20)

---

*L'épreuve comporte deux parties indépendantes qui devront être rédigées sur deux copies séparées : **copie 1 pour la première partie ; copie 2 pour la deuxième partie**. Chaque feuille devra comporter l'indication claire "**copie 1**" ou "**copie 2**" puisque les feuilles relatives à chacune des parties seront ramassées séparément.*

PREMIERE PARTIE

(Note sur 5)
La durée conseillée pour cette partie est de 45 mn

Le but de ce problème est de comparer deux algorithmes de division. On désire manipuler des nombres réels de [0,1[avec une grande précision. Pour ceci, le réel x qui s'écrit sur N chiffres $0.x_1x_2x_3 \ldots x_N$ en base 2 (c'est-à-dire $x = \sum_{i=1}^{N} x_i 2^{-i}$, où x_i vaut 0 ou 1) sera représenté par un tableau T de N éléments de type *integer*, tel que $T[i] = x_i$. On utilisera donc des variables de type suivant :

```
const N = 1000;  (* par exemple, mais N pourra être encore plus grand *)
type grandreel = array [1..N] of integer
```

Question 1 :
a. Ecrire une fonction PASCAL :

plusgrand (x,y : grandreel) : **boolean**;

telle que plusgrand (x,y) = $\begin{cases} \textbf{true si } x \geq y \\ \textbf{false si } x < y \end{cases}$

b. Donner une procédure PASCAL :

foisdeux (x : grandreel; **var** y : grandreel; **var** tropgrand : **boolean**);

telle que :

- si $2x \geq 1$ alors tropgrand = **true**
- si $2x < 1$ alors tropgrand = **false** et $y = 2x$.

Tournez la page S.V.P.

c. On suppose que l'on dispose d'une procédure PASCAL :

diff (x,y : grandreel; var z: grandreel)

telle que si $x \geq y$, alors $z = x-y$. En déduire une procédure PASCAL :

itère (x,y : grandreel; var z : grandreel; var tropetit, erreur : boolean)

telle que **si** $x < \frac{1}{2}$, alors :

- erreur = **false**
- tropetit = **true** et $z = 2x$ si $2x - y < 0$
- tropetit = **false** et $z = 2x-y$ si $2x - y \geq 0$

et **si** $x \geq \frac{1}{2}$ alors erreur = **true**

Question 2 :

Les variables x et y sont de type *grandreel*. On suppose $0 \leq x < y < \frac{1}{2}$. Montrer que la suite $x^{(i)}$ de réels et la suite q_i de chiffres de base 2 définies par :

$$x^{(0)} = x$$
$$x^{(i+1)} = 2x^{(i)} - q_{i+1}y$$

où $q_{i+1} = \begin{cases} 0 \text{ si } 2x^{(i)} < y \\ 1 \text{ si } 2x^{(i)} \geq y \end{cases}$ vérifient :

- $0 \leq x^{(i)} < y$ pour tout i
- $\sum_{i=1}^{\infty} q_i 2^{-i} = \frac{x}{y}$ (autrement dit : $\frac{x}{y}$ s'écrit $0.q_1q_2q_3q_4 \ldots q_iq_{i+1} \ldots$ en base 2)

En déduire une procédure PASCAL qui, en fonction de deux éléments x et y de type *grandreel*, tels que $x < y < \frac{1}{2}$, calcule un élément z de type *grandreel* , formé des N premiers chiffres (en base 2) de $\frac{x}{y}$. Si la contrainte $x < y < \frac{1}{2}$ n'est pas satisfaite, indiquer sommairement (on ne demande pas l'écriture d'une procédure) comment on peut s'y ramener.

Question 3 :

On désire calculer par une autre méthode x/y, où x et y sont de type *grandreel*. On suppose que y est supérieur ou égal à 1/2, et que x est strictement inférieur à y. On pose $\varepsilon = 1-y$.

a. montrer la relation :

$$x\,(1+\varepsilon)(1+\varepsilon^2)(1+\varepsilon^4)(1+\varepsilon^8)(1+\varepsilon^{16}) \ldots (1+\varepsilon^{(2^i)}) = \frac{x}{y}\,(1 - \varepsilon^{(2^{i+1})}) \quad (\mathcal{R})$$

On suppose que l'on dispose des procédures suivantes :

Procedure unmoins (x : grandreel; **var** y : grandreel; **var** tropgrand : boolean) qui donne :

- Si $x \neq 0$ alors *tropgrand* = **false** et $y = 1-x$.
- Si $x = 0$ alors *tropgrand* = **true**.

procedure somme (x,y : grandreel; **var** z : grandreel; **var** tropgrand : boolean) qui donne :

- Si $x + y < 1$ alors *tropgrand* = **false** et $z = x+y$
- Si $x + y \geq 1$ alors *tropgrand* = **true**.

On suppose également que l'on dispose de la procédure :

procedure mult (x,y : grandreel; **var** z : grandreel)

qui place dans *z* le produit de *x* par *y*. *On fera comme si cette opération était exacte* (en réalité, le produit de 2 nombres de N chiffres ne tient pas forcément exactement sur N chiffres. Tenir compte de ce fait entrainerait une étude un peu plus délicate, mais la conclusion finale resterait la même), et on suppose connue la constante K égale au plus petit entier supérieur ou égal à $\log_2(N)$. Déduire de la relation ($\mathcal{R}$) une procédure donnant $\frac{x}{y}$ avec N chiffres significatifs de base 2, c'est à dire une procédure fournissant un résultat z vérifiant :

$$\left| z - \frac{x}{y} \right| < 2^{-N}$$

b. On supposera qu'une bonne estimation du temps d'exécution d'une procédure est donnée par le nombre d'**affectations** (c'est à dire le nombre d'instructions de la forme "a := b" portant sur des variables de type **integer** ou **boolean**) effectuées. Par exemple si c et d sont de type **integer**, la suite d'instructions:

for i := 1 **to** M **do if** (a [i] >= b) **then** c := c + d **else** c := -1

conduira à l'exécution de M affectations (le ":=" de **for** i := 1 **to** M ne sera pas compté). On comptera l'affectation "x := y" où x et y sont de type *grandreel* comme N affectations (car elle est équivalente à "**for** i := 1 **to** N **do** x[i] := y[i]")

- On suppose que la procédure *diff* de la **question 1** entraîne l'éxécution de $C_1 N$ affectations ($C_1 \in \mathbb{N}^*$). Donner en fonction de N un minorant et un majorant du nombre d'affectations effectuées lors de l'exécution de la procédure de division de la question 2.
- La procédure *mult* utilise un algorithme dû à Schönhage et Strassen, qui permet d'effectuer des multiplications en utilisant au plus $C_2.N.\log_2(N).\log_2(\log_2 N))$ affectations ($C_2 > 0$). Les procédures *somme* et *unmoins* utilisent respectivement $C_3 N$ et $C_4 N$ affectations, ($C_3 > 0$, $C_4 > 0$). En déduire un majorant du nombre d'affectations utilisées par la procédure de division de la **question 3.a.** Comparer les deux algorithmes de division obtenus.

Tournez la page S.V.P.

ENS LYON INFORMATIQUE

DEUXIEME PARTIE

(Note sur 15)

Introduction et Notations : L'objet de ce problème est d'étudier un procédé permettant de mémoriser une famille donnée de vecteurs à composantes discrètes.

On note H la fonction définie sur $\mathbb{R}$ par $H(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ -1 & \text{sinon} \end{cases}$

Si E est un ensemble fini, CardE désigne son cardinal. On se donne un entier positif N et on appellera vecteur état à N composantes un vecteur dont les N composantes ont pour valeur -1 ou +1. On désignera par $\mathfrak{E}_N$ l'ensemble de ces états :

$\mathfrak{E}_N = \{x = (x_I), I = 1, 2 ..., N \mid x_I \in \{-1, +1\}\}$

On appellera x_I l'état du nœud I.

Etant donnés n états $x^1, x^2, ..., x^n$, on note X la matrice à N lignes et n colonnes formée par les vecteurs x^i : $X = (x^1, x^2, ..., x^n)$.

On construit les coefficients $c_{IJ}(X)$ en posant :

$$c_{IJ}(X) = \sum_{i=1}^{n} x_I^i x_J^i \quad , (I,J) \in [1,N]^2,$$

et on définit l'application K_X de $\mathfrak{E}_N$ dans $\mathfrak{E}_N$ en posant :

$$[K_X(y)]_I = H\left(\sum_{J=1}^{N} c_{IJ}(X)\, y_J \right) \quad , I = 1, ..., N.$$

On cherche à déterminer dans quelle mesure on a : $K_X(x^i) = x^i$, $i = 1, 2 ..., n$.

On dira dans ce cas-là que l'on a mémorisé les états x^i, $i \in [1,n]$ par l'intermédiaire de l'application K_X.

1°. On se place dans le cas $N = 5$ et on considère les $n = 5$ états obtenus en prenant des composantes égales à -1, à l'exception d'une seule :

$$x_I^i = \begin{matrix} -1 & \text{si } i \neq I \\ 1 & \text{si } i = I \end{matrix} \quad , \quad I \in [1,5], i \in [1,5] \ .$$

Calculer dans ce cas la matrice (c_{IJ}) puis les états $K_X(x^i)$, $i = 1, 2, 3, 4, 5$.

2°. Si x et y sont deux états de $\mathbb{R}^N$, on définit $\langle x,y \rangle$ comme étant le produit scalaire usuel et on dira que x et y sont orthogonaux si $\langle x,y \rangle = 0$.

2°.a) Montrer qu'on a : $K_X(x^i) = H\left(\sum_{j=1}^{n} \langle x^i, x^j \rangle\, x^j \right)$ *

2°.b) En déduire que si la matrice de terme général $a_{ij} = \langle x^i, x^j \rangle$, $(i,j) \in [1,n]^2$, vérifie

$$\forall i \in [1,n] \qquad |a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

alors on a

$$\forall i \in [1,n] \qquad K_X(x^i) = x^i.$$

* Lire $\left[K_X(x^i)\right]_I = H\left(\sum_{j=1}^{n} \langle x^i, x^j \rangle\, x_I^j \right)$

3°. On se fixe maintenant N, n, un indice I_o dans [1,N], un indice i_o dans [1,n], et on cherche à dénombrer les familles $\{x^i\}$ de n états à N composantes telles que :

$$K_X(x^{i_o})_{I_o} = x^{i_o}_{I_o} .$$

Dans ce but, on pose :

$$A = \left\{ \{x^i\}_{i=1}^n \in (\mathfrak{E}_N)^n \mid K_X(x^{i_o})_{I_o} = x^{i_o}_{I_o} \right\}$$

et $$m(A) = \frac{CardA}{2^{Nn}}$$

On définit sur $\mathfrak{E}_N$ la distance d par la formule : $d(x,x') = Card\ \{I \in [1,N] \mid x_I \neq x'_I\}$

3°.a) Montrer que pour $x, x' \in \mathfrak{E}_N$ on a : $\langle x,x' \rangle = N - 2\, d(x,x')$

3°.b) En déduire que

$$A \supset \left\{ \{x^i\}_{i=1}^n \mid \sum_{\substack{k=1 \\ k \neq i_o}}^{n} [N - 2\, d(x^k, x^{i_o})]\ x^k_{I_o} x^{i_o}_{I_o} > -N \right\}$$

3°.c) Montrer que l'application définie de $(\mathfrak{E}_N)^n$ dans lui-même par $x^{i_o} \to x^{i_o}$

et, si $i \neq i_o$, $$x^i \to y^i \text{ avec } \begin{cases} y^i_I = - x^i_I \text{ si } I \neq I_o \\ y^i_{I_o} = x^i_{I_o} \end{cases}$$

envoie le complémentaire de A dans $(\mathfrak{E}_N)^n$ dans A.

En déduire que $m(A) \geq \frac{1}{2}$ pour tout $(n,N) \in \mathbb{N}^2$.

3°.d) On se donne dans cette question un entier P compris entre 2 et N/2 et on définit l'ensemble B des familles de n états tels qu'au moins P coordonnées de l'état x^{i_o} sont préservées par l'application K_X associée :

$B = \{ \{x^i\}_{i=1}^n \in (\mathfrak{E}_N)^n \mid \exists$ P indices 2 à 2 distincts $I_1, I_2, ..., I_P$ vérifiant $K_X(x^{i_o})_I = x^{i_o}_I$

pour tout $I \in \{I_1, I_2, ..., I_P\} \}$.

Soit B' le complémentaire de B dans $(\mathfrak{E}_N)^n$. Montrer que si $\{x^i\}_{i=1}^n \in B'$, alors il existe P indices 2 à 2 distincts $I_1, ..., I_P$ dans [1,2P - 1] tels que $K_X(x^{i_o})_I \neq x^{i_o}_I$ pour tout $I \in \{I_1, ..., I_P\}$. En déduire que $m(B) \geq \frac{1}{2}$ si $n < 1 + \frac{N}{2P-2}$.

Indication : On pourra utiliser l'application de $(\mathfrak{E}_N)^n$ dans lui-même définie par $x^{i_o} \to x^{i_o}$

et, si $i \neq i_o$, $$x^i \to y^i \text{ avec } \begin{cases} y^i_I = - x^i_I & \text{si } I \notin [1, 2P-1] \\ y^i_I = x^i_I & \text{sinon .} \end{cases}$$

3°.e) Interpréter les résultats des deux questions précédentes en termes de capacité de mémorisation des applications K_X .

4°. On s'intéresse maintenant à l'attractivité des états x^i lorsque ceux-ci sont points fixes de l'application K_X .

Tournez la page S.V.P.

ENS LYON INFORMATIQUE

4°.a) On reprend l'exemple traité dans la question 1° et on considère l'état y défini par :

$$y_1 = y_5 = 1 \quad ; \quad y_2 = y_3 = y_4 = -1$$

Calculer $K_X(y)$.
Refaire le calcul dans le cas où on construit K_X à partir de $X = (x_1, x_2, x_3, x_4)$.(c'est à dire qu'on ne conserve que les quatre premiers états).
Qu'observe-t-on ?

4°.b) Dans le cas général, soit $\{x^i\}$ une famille de n états de $\mathfrak{E}_N$, K_X l'application associée et $\{y^i\}$ une deuxième famille de n états.
Montrer que si la matrice de terme général $b_{ij} = \langle x^i, y^j \rangle$ vérifie

$$\forall\, i \in [1,n] \qquad b_{ii} > \sum_{j \neq i} |b_{ji}|$$

alors on a

$$\forall\, i \in [1,n] \qquad K_X(y^i) = x^i .$$

4°.c) En déduire que, si on suppose les x^i orthogonaux 2 à 2, on a, pour un indice i_o dans [1,n],

$$K_X(y) = x^{i_o}$$

dès que

$$d(y, x^{i_o}) < \frac{N}{2n} .$$

Montrer par un exemple avec $N = 4$ que l'inégalité stricte ci-dessus ne peut pas en général être remplacée par une inégalité large.