e3a

CONCOURS ARTS ET MÉTIERS ParisTech - ESTP - ARCHIMEDE

Épreuve d'Informatique MP

Durée 3 h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est interdit.

AVERTISSEMENT

La **présentation**, la lisibilité, l'orthographe, la qualité de la **rédaction, la clarté et la précision** des raisonnements entreront pour une **part importante** dans **l'appréciation des copies**. En particulier, les résultats non encadrés et non justifiés ne seront pas pris en compte.

Indiquer en tête de copie le langage de programmation, Caml ou Pascal, choisi pour l'ensemble du sujet.

- L'épreuve est constituée de cinq exercices totalement indépendants.
- Pour les candidats composant avec Pascal, on suppose disposer de deux types de listes : le type listes d'entiers **Liste** et le type liste de liste d'entiers **Lliste**. La liste vide sera représentée pour les deux types par **nil** et on suppose disposer des fonctions suivantes :
  - **cons** prenant en argument un entier x et une liste d'entiers l et renvoyant une liste d'entiers dont la tête est x et la queue l.
  - **lcons** prenant en argument une liste d'entiers x et une liste de listes d'entiers l et renvoyant une liste de listes d'entiers dont la tête est x et la queue l.
  - **tete** (resp. **ltete**) qui renvoie la tête d'une liste d'entiers (resp. d'une liste de listes d'entiers).
  - **queue** (resp. **lqueue**) qui renvoie la queue d'une liste d'entiers (resp. d'une liste de listes d'entiers).

Tournez la page S.V.P.

1 Décompositions rationnelles

On considère un rationnel $x \in]0,1[$. On note $x = \frac{p}{q}$ avec $q \geqslant 2$ et $0 < p < q$. On s'intéresse au problème suivant : peut-on trouver des entiers $2 \leqslant a_1 < a_2 < ... < a_n$ de sorte que :

$$x = \frac{1}{a_1} + \frac{1}{a_2} + ... + \frac{1}{a_n} ?$$

Par exemple, on a :

$$\frac{3}{8} = \frac{1}{3} + \frac{1}{24}$$

On considère pour cela l'algorithme suivant :

```
decompose p q=
      res:=[]
      fonction aux a b=
            reste:=b mod a
            quotient:=b/a
              si reste=0 alors add (quotient,res)
                  sinon add (quotient+1,res);aux (a-reste) (b*(quotient+1))
              fin si
      fin fonction
      aux p q
      renvoyer res
```

où $b \bmod a$ et b/a désignent le reste et le quotient de la division euclidienne de b par a, $[\,]$ désigne la liste vide et la fonction *add* permet d'ajouter un élément en tête d'une liste.

1. Détailler l'exécution de l'algorithme sur les fractions $\frac{5}{8}$ et $\frac{3}{7}$.
2. Prouver que l'algorithme ci-dessus termine et majorer sa complexité à l'aide de p et q.
3. Prouver que l'algorithme est correct.
4. On note, pour $n \geqslant 1$ et $k \geqslant 1$, $\alpha_n = n! + 1$ et $\beta_{k,n} = k(n!) + 1$.
   (a) Écrire la division euclidienne de $\beta_{k,n}$ par n. (On pourra distinguer les cas $n = 1$ et $n \neq 1$.)
   (b) Prouver que l'exécution de l'algorithme sur $p = n$ et $q = \beta_{k,n}$ renvoie une liste de longueur n.
   (c) En déduire que la décomposition renvoyée par l'algorithme sur $p = n$ et $q = \alpha_n$ est de longueur n.
   (d) Que peut-on en déduire quant à la complexité de l'algorithme ?

2 Calcul des termes d'une suite double

On considère la suite double (S_n^p) définie pour n et p entiers naturels par :

$$S_0^0 = 1 \quad S_0^p = 0 \text{ si } p \geqslant 1 \quad S_n^0 = 0 \text{ si } n \geqslant 1 \text{ et } \forall n \geqslant 1, \forall p \geqslant 1, S_n^p = p(S_{n-1}^{p-1} + S_{n-1}^p)$$

1. Proposer un programme récursif prenant n et p en argument et renvoyant la valeur de S_n^p. L'usage de variables auxiliaires n'est pas autorisé dans cette question.
2. On s'intéresse à la complexité en nombre d'additions et de multiplications nécessaires au calcul de S_n^p par le programme proposé à la question précédente. On note $C(n,p)$ le nombre d'opérations nécessaires au calcul de S_n^p et $C_n = \sup_{p \in \mathbb{N}} C(n,p) \in \mathbb{N} \cup \{+\infty\}$.
   (a) Déterminer suivant $p \in \mathbb{N}$ les valeurs de $C(0,p)$, de $C(1,p)$ et de $C(2,p)$ et en déduire les valeurs de C_0, C_1 et C_2.
   (b) Pour $n \in \mathbb{N}$, justifier que C_n est fini et montrer que $C_n \leqslant 2^{n+1}$.
   (c) Pour $p \geqslant n \geqslant 1$, obtenir l'existence d'une constante $\beta > 1$ telle que $C(n,p) \geqslant \beta^n$.
   (d) Comment qualifier la complexité de votre programme ?
3. En vous aidant d'un tableau de longueur $p+1$, écrire une fonction calculant S_n^p en $O(np)$ opérations.

3 Énumération des parties d'un ensemble

Pour $n \in \mathbb{N}^*$, on note E_n l'ensemble $[\![1, n]\!]$ et on pose $E_0 = \emptyset$. Une partie A de l'ensemble E_n est représentée par une liste $[a_1, ..., a_p]$ où :

$$A = \{a_1, ..., a_p\} \text{ et } a_1 < a_2 < ... < a_p$$

1. (a) Soit A une partie de E_n représentée par une liste l et soit $x \in E_n$. Écrire des fonctions **add** et **sub** prenant en argument l et x et qui renvoient respectivement les listes représentant les parties $A \cup \{x\}$ et $A \setminus \{x\}$.

   (b) Écrire des fonctions récursives **reunion** et **intersection** prenant en argument deux listes l_1 et l_2 représentant deux parties A_1 et A_2 de l'ensemble E_n et renvoyant respectivement la réunion et l'intersection de A_1 et A_2.

2. (a) Écrire une fonction récursive **parties** prenant n et p en arguments et renvoyant la liste de toutes les parties à p éléments de l'ensemble E_n. Par exemple, lorsque $n = 2$ et $p = 1$, le résultat renvoyé est : $[[1], [2]]$; lorsque $n = 2$ et $p = 0$, le résultat renvoyé est $[[\,]]$. On pourra remarquer que l'ensemble des parties à p éléments se partitionne en l'ensemble des parties à p éléments contenant n et l'ensemble des parties à p éléments ne contenant pas n.

   (b) Détailler l'exécution de votre algorithme sur les entiers $n = 3$ et $p = 2$.

4 Fonctions booléennes

Soit $n \in \mathbb{N}^*$. On convient de confondre les booléens «vrai» et «faux» avec les entiers 1 et 0. On appelle fonction booléenne toute application :

$$\begin{array}{rccc} f : & \{0,1\}^n & \to & \{0,1\} \\ & (x_1, ..., x_n) & \mapsto & f(x_1, ..., x_n) \end{array}$$

4.1 Un opérateur universel

1. Soit f une fonction booléenne. Justifier l'égalité :

   $$\forall (x_1, ..., x_n) \in \{0,1\}^n, f(x_1, ..., x_n) = [x_n \implies f(x_1, ..., x_{n-1}, 1)] \wedge [(\neg x_n) \implies f(x_1, ..., x_{n-1}, 0)]$$

2. Soit $n \in \mathbb{N}^*$. Montrer que toute fonction booléenne de $\{0,1\}^n$ dans $\{0,1\}$ peut s'écrire à l'aide des opérateurs $\neg$, $\wedge$ et $\vee$ et des variables x_1, x_2,..., x_n.
3. On pose $A \uparrow B = (\neg A) \vee (\neg B)$. Montrer que toute fonction booléenne peut s'écrire uniquement à l'aide de l'opérateur $\uparrow$ et des variables x_1, x_2,..., x_n.

4.2 Formules croissantes

On munit $\{0,1\}^n$ de l'ordre produit :

$$(x_1, ..., x_n) \leq (y_1, ..., y_n) \Leftrightarrow \forall i \in [\![1, n]\!], x_i \leqslant y_i$$

On dit qu'une fonction booléenne f est croissante lorsque :

$$\forall (x, y) \in (\{0,1\}^n)^2, x \leq y \implies f(x) \leqslant f(y)$$

1. Donner un exemple de fonction f telle que ni f, ni $\neg f$ ne soit croissante. (On justifiera soigneusement le contre-exemple.)
2. Peut-on affirmer que si $f : \{0,1\}^n \to \{0,1\}$ est croissante alors $f_0 : (x_1, ..., x_{n-1}) \mapsto f(x_1, ..., x_{n-1}, 0)$ et $f_1 : (x_1, ..., x_{n-1}) \mapsto f(x_1, ..., x_{n-1}, 1)$ sont croissantes ? (On donnera une preuve ou un contre-exemple.)
3. Si f est croissante, montrer que $f_1 = f_0 \vee f_1$.
4. Montrer que la fonction f est croissante si et seulement si elle est équivalente à une formule écrite à l'aide des opérateurs $\wedge$ et $\vee$, des variables x_1, x_2,..., x_n et des fonctions constantes à 0 et 1.

Tournez la page S.V.P.

5 Préfixes et suffixes

Soit Σ un alphabet fini. Soit $u \in \Sigma^*$. Un mot $v \in \Sigma^*$ est un préfixe de u lorsqu'il existe $w \in \Sigma^*$ tel que $u = vw$ et un mot $w \in \Sigma^*$ est un suffixe de u lorsqu'il existe $v \in \Sigma^*$ tel que $u = vw$. Pour la suite, on pose : $\Sigma = \{a, b\}$ et $u = baabbaa$.

1. Dresser la liste des préfixes de u ainsi que la liste de ses suffixes.
2. Représenter graphiquement un automate déterministe reconnaissant exactement l'ensemble des préfixes de u.
3. (a) Représenter graphiquement un automate *non déterministe* simple reconnaissant exactement l'ensemble des suffixes de u.

   (b) Représenter graphiquement le résultat de l'application de l'algorithme de déterminisation sur l'automate obtenu à la question précédente.