NG92

e3a

CONCOURS ENSAM - 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.

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

Lorsqu'un programme est donné dans l'énoncé, une version en Caml et une version en Pascal sont fournies : ne considérer que celle écrite dans le langage que vous avez choisi. La présentation et la qualité de la rédaction de votre composition seront évaluées. La clarté et la concision des programmes rédigés seront également prises en compte.

1 Sommes d'entiers

Dans cet exercice, les arguments $(a_1, a_2, ..., a_p)$ *et* $((a_1, max_1), (a_2, max_2), ..., (a_p, max_p))$ *des fonctions seront, au choix, considérés comme des tableaux ou des listes (contenant des entiers ou des couples d'entiers).*

On considère des suite finies d'entiers naturels non nuls : $0 < a_1 < a_2 < ... < a_p$. On s'intéresse au problème suivant : étant donné un entier $n \in \mathbb{N}$, peut-on écrire :

$$n = \sum_{k=1}^{p} \lambda_k a_k \text{ où } (\lambda_1, ..., \lambda_p) \in \mathbb{N}^p \text{ ?}$$

1. On se place dans le cas où $p = 2$ et $a_1 = 5$ et $a_2 = 7$. Étudier le problème pour n entre 10 et 14.
2. Écrire un programme prenant en argument n et les éléments a_1, a_2,..., a_p et renvoyant le booléen «vrai» si une telle décomposition existe et «faux» sinon.
3. On ajoute la contrainte suivante : on se donne des entiers naturels non nuls max_1, max_2,..., max_p et on impose d'avoir les inégalités suivantes : $0 \leqslant \lambda_k \leqslant max_k$ pour tout $k \in [1, p]$.
   (a) Écrire un programme prenant en argument n et les couples (a_1, max_1), (a_2, max_2),..., (a_p, max_p) et renvoyant le booléen «vrai» si une telle décomposition existe et «faux» sinon.
   (b) Écrire un programme prenant en argument n et les couples (a_1, max_1), (a_2, max_2),..., (a_p, max_p) et renvoyant le nombre de telles décompositions.

2 Autour de la racine carrée entière

On s'intéresse au problème suivant : étant donné un entier $n \geqslant 1$, on souhaite déterminer un entier positif s_n de sorte que s_n soit «proche» de $\sqrt{n}$. Nous étudierons différentes approches pour résoudre ce problème. On se donne les contraintes suivantes :

1. On ne travaillera qu'avec des entiers relatifs ;
2. Les opérations arithmétiques autorisées sur les entiers sont :
   (a) l'addition de deux entiers a et b : `a + b` ;
   (b) la soustraction de deux entiers a et b : `a - b` ;
   (c) la multiplication de deux entiers a et b : `a * b` ;
   (d) la division euclidienne d'un entier positif a par un entier strictement positif b : le reste de la division étant obtenu par `a` mod `b` et le quotient par `a/b` en Caml et par `a div` b en Pascal.

Dans la suite, si a et b sont deux entiers avec $b \neq 0$, l'écriture $\frac{a}{b}$ désigne le quotient **rationnel** de a par b ; si x est un réel, $\lfloor x \rfloor$ désigne la partie entière de x et $\lceil x \rceil$ désigne le plafond de x : le plus petit entier supérieur ou égal à x.

2.1 Algorithme naïf

Dans cette partie, étant donné un entier $n \geqslant 1$, on cherche à déterminer le plus grand entier s_n tel que $s_n^2 \leqslant n$. Pour cela, on passe en revue les entiers 1, 2, 3,... jusqu'à trouver s_n.

1. Écrire une fonction `isqrt_naif` prenant n en argument et renvoyant la valeur de s_n en utilisant le principe décrit ci-dessus.
2. Déterminer la complexité de votre programme.

2.2 Une méthode dichotomique

Dans cette partie, on reprend le problème de la partie précédente : étant donné un entier $n \geqslant 1$, on cherche à déterminer le plus grand entier s_n tel que $s_n^2 \leqslant n$. Afin d'obtenir une méthode plus efficace, on considère les programmes suivants :

```
let isqrt n=
    let x=ref 1 in
       let y=ref n in
          while((!y)-(!x)>1) do
             let z=((!x)+(!y))/2 in
                 if z*z>n then
                      y:=z
                 else
                      x:=z
          done;
    !x;;
```

```
Function isqrt(n: longint) : longint;
Var x,y,z : longint;
Begin
   x:=1;y:=n;
   While (y-x>1) Do
   begin
      z:=(x+y) div 2;
      if z*z>n then
         y:=z
      else
         x:=z
   end;
   isqrt:=x;
End;
```

Si $k \geqslant 1$, on note x_k et y_k les valeurs des variables x et y à la fin de la k-ième itération de la boucle `while`. On convient que $x_0 = 1$ et $y_0 = n$.

1. Expliciter les exécutions du programme ci-dessus sur l'entier 15. (On donnera en particulier les valeurs successives des variables x, y et z.)
2. Démontrer la terminaison du programme.
3. Justifier la correction du programme.
4. (a) Prouver l'inégalité suivante :
$$y_k - x_k \leqslant \frac{y_0 - x_0 - 1}{2^k} + 1$$
   (b) En déduire une majoration de la complexité du programme.
5. Réécrire la fonction `isqrt` en utilisant de la récursivité.

2.3 Circuit logique

On se donne un entier $x \in [0, 15]$ dont le développement binaire est $x = a_3a_2a_1a_0$. Par exemple, l'entier 7 s'écrit 0111. Le développement binaire de l'entier $y = \lfloor \sqrt{x} \rfloor$ s'écrit alors $y = b_1b_0$.

1. *Pour les deux questions suivantes, on veillera à donner un résultat le plus simple possible.*
   (a) Donner une formule booléenne exprimant b_1 en fonction de a_0, a_1, a_2 et a_3.
   (b) Donner une formule booléenne exprimant b_0 en fonction de a_0, a_1, a_2 et a_3.
2. On souhaite construire des circuits logiques implémentant ces deux fonctions. On dispose seulement de porte de type «ou» et de type «non» représentées dans la figure 1.
   (a) Proposer un circuit implémentant la fonction b_1.
   (b) Proposer un circuit implémentant la fonction b_0 n'utilisant au plus que quatre portes de type «ou» et quatre portes de type «non».

FIGURE 1 – Portes «ou» et «non»

2.4 Une méthode Babylonienne

On se donne un entier $a \geqslant 1$ et on considère la suite, définie par $u_0 = 1$ et, pour $n \geqslant 0$:

$$u_{n+1} = \left\lceil \frac{\left\lfloor \frac{a-1}{u_n} \right\rfloor + u_n}{2} \right\rceil$$

On admet que la suite (u_n) est correctement définie et qu'elle converge vers l'unique entier le plus proche de $\sqrt{a}$. Par exemple, l'entier le plus proche de $\sqrt{2} = 1,414...$ est 1 et l'entier le plus proche de $\sqrt{7} = 2,645...$ est 3.

1. Écrire une fonction prenant a supposé supérieur ou égal à 1 et calculant l'entier le plus proche de $\sqrt{a}$. On détaillera la manière dont est effectué le calcul des parties entières et des plafonds.
2. Déterminer les valeurs de la suite (u_n) lorsque a vaut 13 et donner la valeur de l'unique entier le plus proche de $\sqrt{13}$.

3 Coefficients binomiaux

Dans cet exercice, on supposera que le type **int** *permet de représenter des entiers arbitrairement grands.*

On se donne deux entiers $k \in \mathbb{Z}$ et $n \in \mathbb{N}$. Le coefficient binomial associé à ces deux entiers est :

$$\binom{n}{k} = \begin{cases} \dfrac{n!}{(n-k)!k!} & \text{si } 0 \leqslant k \leqslant n \\ 0 & \text{sinon} \end{cases}$$

On a alors la formule de Pascal :

$$\forall n \geqslant 1, \forall k \in \mathbb{Z}, \binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

1. *Dans cette question, on n'utilisera ni liste, ni tableau et la seule opération arithmétique autorisée est l'addition ; on n'effectuera donc aucune multiplication.* Écrire une fonction récursive simple calculant le coefficient $\binom{n}{k}$.
2. On pose $n = 1000$ et on souhaite construire un tableau t de longueur $n+1$ tel que si $k \in [0, n]$, $t[k]$ contient la valeur du coefficient $\binom{n}{k}$.
   (a) Expliquer brièvement pourquoi il n'est pas possible d'utiliser la fonction de la question précédente.

(b) Écrire un programme renvoyant le tableau convenablement rempli.

3. *Cette question est indépendante des questions précédentes. Il est demandé de ne pas utiliser de tableau et de ne pas calculer* $\binom{n}{k}$. Un théorème de Lucas assure que si k et n sont des entiers naturels, alors, en écrivant k et n en base 2 :

$$k = k_p 2^p + k_{p-1} 2^{p-1} + ... + k_1 2^1 + k_0 2^0 \text{ et } n = n_p 2^p + n_{p-1} 2^{p-1} + ... + n_1 2^1 + n_0 2^0$$

on a :

$$\binom{n}{k} \equiv \prod_{i=0}^{p} \binom{n_i}{k_i} \mod [2]$$

Écrire un programme prenant en argument deux entiers k et n supposés naturels et renvoyant le booléen «vrai» si $\binom{n}{k}$ est pair et le booléen «faux» sinon.

4 Minimisation d'automates

Dans cette partie, on considère l'alphabet $\Sigma = \{a, b\}$. On rappelle les définitions suivantes :

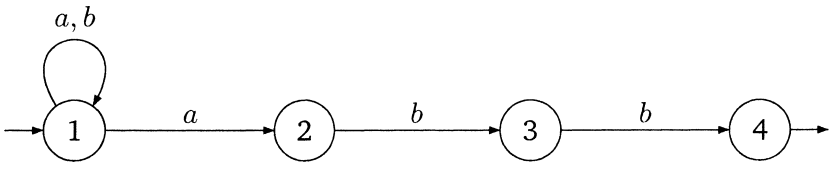
- un automate déterministe sur l'alphabet Σ est un quadruplet $\mathcal{A} = (Q, \delta, i_0, F)$ où :
  - Q est un ensemble fini d'états
  - $\delta : Q \times \Sigma \to Q$ est une application appelée fonction de transition
  - $i_0 \in Q$ est l'état initial et
  - $F \subseteq Q$ est l'ensemble des états finals.
- un automate sur l'alphabet Σ est un quadruplet $\mathcal{A} = (Q, T, I, F)$ où :
  - Q est un ensemble fini d'états
  - $T \subset Q \times \Sigma \times Q$ est l'ensemble des transitions
  - $I \subseteq Q$ est l'ensemble des états initiaux et
  - $F \subseteq Q$ est l'ensemble des états finals.

Le terme «automate» désigne dans la suite un automate pouvant être non déterministe. Si $\mathcal{A}$ est un automate, on note $d(\mathcal{A})$ l'automate déterministe obtenu à partir de $\mathcal{A}$ par l'algorithme habituel de déterminisation. Si $\mathcal{A}$ est un automate, l'automate miroir de $\mathcal{A}$ est l'automate $m(\mathcal{A})$ défini par : $m(\mathcal{A}) = (Q_m, T_m, I_m, F_m)$ où :

- $Q_m = Q$;
- T_m est défini par : $(q, s, q') \in T_m \Leftrightarrow (q', s, q) \in T$;
- $I_m = F$ et
- $F_m = I$.

Finalement, si $\mathcal{A} = (Q, T, I, F)$ désigne un automate et si q est un état de $\mathcal{A}$, le langage reconnu par l'état q est le langage reconnu par l'automate $\mathcal{A}_q = (Q, T, \{q\}, F)$.

Dans la suite $\mathcal{A}$ désigne l'automate suivant :



On pose $\mathcal{A}' = d(m(\mathcal{A}))$ et $\mathcal{A}'' = d(m(\mathcal{A}'))$.

1. Donner une expression rationnelle du langage reconnu par l'automate $\mathcal{A}$.
2. (a) Représenter graphiquement l'automate $\mathcal{A}'$.
   (b) Représenter graphiquement l'automate $\mathcal{A}''$. (*Indication : l'automate $\mathcal{A}''$ a quatre états.*)
3. Si $\mathcal{L}$ est un langage, on définit le langage miroir $m(\mathcal{L})$ de $\mathcal{L}$ par :

$$s_1 s_2 ... s_n \in m(\mathcal{L}) \Leftrightarrow s_n s_{n-1} ... s_2 s_1 \in \mathcal{L}$$

   (a) Si $\mathcal{L}$ est un langage reconnaissable, montrer que $m(\mathcal{L})$ est également reconnaissable.
   (b) Prouver que si $\mathcal{B}$ est un automate, alors l'automate $d(m(d(m(\mathcal{B}))))$ est un automate déterministe qui reconnaît le même langage que l'automate $\mathcal{B}$.
4. (a) Démontrer que les états de l'automate $\mathcal{A}''$ reconnaissent des langages distincts.
   (b) Déduire de ce qui précède qu'il n'existe aucun automate déterministe reconnaissant le même langage que $\mathcal{A}$ et ayant strictement moins d'états que $\mathcal{A}''$.