Option Informatique en Spé MP et MP*
DS du mercredi 21 mars 2007 : le corrigé

▶ J'espère que ce problème vous a semblé intéressant, malgré plusieurs imperfections : erreur dans les formules définissant $\psi(k)$ et $\overline{\psi}(k)$, définition erronée de la méthode d'implantation qualifiée de «lexicographique».

Partie 2

Question 1 • La liste $(\mathbf{0} \cdot \ell)$ vérifie la condition de l'énoncé ; par miroir, il en est de même de la liste $(\mathbf{1} \cdot \ell^R)$. Enfin, le dernier mot de $(\mathbf{0} \cdot \ell)$ et le premier mot de $(\mathbf{1} \cdot \ell^R)$ ne diffèrent que par leur première lettre. Notons que l'on passe du dernier mot $\mathbf{10}^n$ au premier $\mathbf{0}^{n+1}$ en ne changeant qu'une lettre.

Question 2 • $\Gamma_3 = (\mathbf{000}, \mathbf{001}, \mathbf{011}, \mathbf{010}, \mathbf{110}, \mathbf{111}, \mathbf{101}, \mathbf{100})$.

Question 3 • De par la construction de Γ_{n+1}, nous avons $\gamma(2^n + r) = \mathbf{1} \cdot \gamma(2^n - 1 - r)$, donc $\mathbf{g}(2^n + r) = 2^n + \mathbf{g}(2^n - 1 - r)$.

Question 4 • L'affirmation de l'énoncé est exacte ! Commençons par noter que, avec une récurrence immédiate, $\gamma(2^n - 1) = \mathbf{010}^{n-2}$ et $\gamma(2^n) = \mathbf{110}^{n-2}$.

• Venons-en à ladite assertion ; elle est vraie pour $n = 1$. Supposons-la acquise au rang $n \geqslant 1$, et examinons ce qui se passe dans Γ_{n+1}. Pour les position 0 à $2^n - 1$, la règle proposée s'applique clairement. Soit maintenant $k \in [\![0, 2^n - 2]\!]$: alors $\gamma(2^n + k) = \mathbf{1} \cdot \gamma'(2^n - 1 - k)$ et $\gamma(2^n + k + 1) = \mathbf{1} \cdot \gamma'(2^n - k - 2)$, où $\gamma'(q)$ désigne le q-ième élément de Γ_n. Si $\gamma'(2^n - k - 2)$ contient un nombre pair de $\mathbf{1}$, alors $\gamma'(2^n - 1 - k)$ s'en déduit en changeant le dernier bit : la même règle s'applique donc pour passer de $\gamma(2^n + k)$ à $\gamma(2^n + k + 1)$. Sinon, c'est l'autre règle qui s'applique. Enfin, il reste à examiner le passage de $\gamma(2^n - 1) = \mathbf{0} \cdot \gamma'(2^n - 1) = \mathbf{010}^{n-2}$ à $\gamma(2^n) = \mathbf{110}^{n-2}$: il s'effectue par application de la deuxième règle.

▶ Il y a eu dans le sujet une malheureuse interversion des rôles de a et b dans les quatre questions qui suivent. Il fallait lire $\psi(k) = b_0 b_1 \dots b_{n-2} b_{n-1} \mathbf{0}^\omega$ et $\overline{\psi}(k) = a_0 a_1 \dots$

Question 5 • Nous noterons $\overline{x} = x \oplus \mathbf{1}$. Remarquons que si k est compris entre 0 et $2^n - 1$ inclus, alors l'écriture binaire de $2^n - 1 - k$ se déduit de celle de k par échange des rôles de $\mathbf{0}$ et $\mathbf{1}$; autrement dit, si k s'écrit $x_{n-1} x_{n-2} \dots x_1 x_0$, alors $2^n - k - 1$ s'écrit $\overline{x_{n-1}}\, \overline{x_{n-2}} \dots \overline{x_1}\, \overline{x_0}$. Notons également que $\overline{x} \oplus \overline{y} = x \oplus y$.

La relation proposée est claire lorsque $k = 0$. Supposons-la acquise pour tout $k < 2^n$, et examinons ce qui se passe lorsque $2^n \leqslant k < 2^{n+1}$. L'écriture binaire de $k - 2^n$ (sur n bits) est $b_{n-1} b_{n-2} \dots b_1 b_0$; celle de $\mathbf{g}(n - 2^k)$ est (toujour sur n bits) $\gamma(k - 2^n) = a_{n-1} a_{n-2} \dots a_1 a_0$; donc $\psi(k - 2^n) = b_0 b_1 \dots b_{n-1} \mathbf{0}^\omega$ et $\overline{\psi}(k) = a_0 a_1 \dots a_{n-1} \mathbf{0}^\omega$; ceci, avec $a_j = b_j \oplus b_{j+1}$ pour tout $j \in \mathbb{N}$. L'écriture binaire de k (sur $n + 1$ bits) est $\mathbf{1} b_{n-1} b_{n-2} \dots b_1 b_0$. Avec la remarque faite plus haut et la formule établie à la question 3, celle de $\mathbf{g}(k) = 2^n + \mathbf{g}(k - 2^n - 1)$ est :

$$
\begin{aligned}
\gamma(k) &= \mathbf{1}\gamma(\overline{b_{n-1}}\,\overline{b_{n-2}} \dots \overline{b_1}\,\overline{b_0}) = \mathbf{1}\overline{b_{n-1}}(\overline{b_{n-1}} \oplus \overline{b_{n-2}}) \cdots (\overline{b_1} \oplus \overline{b_0}) \\
&= \mathbf{1}\overline{b_{n-1}}(b_{n-1} \oplus b_{n-2}) \cdots (b_1 \oplus b_0) = \mathbf{1}\overline{b_{n-1}} a_{n-2} \dots a_1 a_0
\end{aligned}
$$

Nous avons donc $\psi(k) = \beta_0 \beta_1 \dots \beta_{n-2} \beta_{n-1} \beta_n \mathbf{0}^\omega$, avec $\beta_j = b_j$ pour $j < n$, $\beta_n = \mathbf{1}$ et $\beta_j = \mathbf{0}$ pour $j > n$; et $\overline{\psi}(k) = \alpha_0 \alpha_1 \dots \alpha_{n-2} \alpha_{n-1} \alpha_n \mathbf{0}^\omega$ avec $\alpha_j = a_j$ pour $j < n - 1$, $\alpha_{n-1} = \overline{b_{n-1}}$ et $\alpha_n = \mathbf{1}$. Pour $j < n - 1$, nous avons $\alpha_j = a_j = b_j \oplus b_{j+1} = \beta_j \oplus \beta_{j+1}$; pour $j > n$, nous avons $\alpha_j = a_j = \mathbf{0}$ et $\beta_j = b_j = \mathbf{0}$; enfin, $\alpha_{n-1} = \overline{b_{n-1}} = b_{n-1} \oplus \mathbf{1} = \beta_{n-1} \oplus \beta_n$ et $\alpha_n = \mathbf{1} = \beta_n + \beta_{n+1}$. Nous avons donc $\alpha_j = \beta_j \oplus \beta_{j+1}$ pour tout $j \in \mathbb{N}$.

Question 6 • La suite de terme général b_j est nulle à partir d'un certain rang n ; il en est donc de même de la suite de terme général a_j, à partir du rang $n + 1$.

Question 7 • La relation est claire si $j > n$; sinon, avec les propriétés de l'addition modulo 2, il vient :

$$b_j \oplus b_{j+1} \oplus \cdots = b_j \oplus b_{j+1} \oplus \cdots \oplus b_n \oplus b_{n+1} = (a_j \oplus a_{j+1}) \oplus (a_{j+1} \oplus a_{j+2}) \oplus \cdots \oplus (a_{n-1} \oplus a_n) = a_j \oplus a_n = a_j$$

Question 8 • Notons $\overline{\psi}(k) = a_0 a_1 \dots$ et $\psi(k) = b_0 b_1 b_2 \dots$; alors $\psi\big(\lfloor k/2 \rfloor\big) = b_1 b_2 \dots$, donc :

$$\psi(k) \oplus \psi\big(\lfloor k/2 \rfloor\big) = (b_0 b_1 b_2 \dots) \oplus (b_1 b_2 \dots) = (b_0 \oplus b_1) \dots (b_1 \oplus b_2) \dots = a_0 a_1 a_2 \dots = \overline{\psi}(k)$$

Partie 3

Question 9 • Immédiat :

n	e_2	e_1	e_0	s_2	s_1	s_0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

Question 10 • Nous obtenons $s_0 = \overline{e_2}\,\overline{e_1}e_0 + \overline{e_2}e_1\overline{e_0} + e_2\overline{e_1}e_0 + e_2e_1\overline{e_0} = \overline{e_1}e_0 + e_1\overline{e_0}$; $s_1 = \overline{e_2}\overline{e_1} + \overline{e_2}e_1$; $s_2 = e_2$.

Question 11 • Dessin à faire !

Question 12 • Si l'on dispose d'une porte «ou exclusif», notée $\oplus$, nous avons $s_0 = e_1 \oplus e_0$ ...

Partie 4

Question 13 • Notons A (resp. B) l'ensemble des indices $i \in [\![1,n]\!]$ tels que $u_i \neq v_i$ (resp. $v_i \neq w_i$) ; ainsi $d(u,v) = |A|$ et $d(v,w) = |B|$. Alors $u_i \neq w_i$ implique $u_i \neq v_i$ ou (inclusif) $v_i \neq w_i$, donc $i \in A \cup B$. Du coup :

$$d(u,w) = \mathrm{Card}\big\{i \in [\![1,n]\!] \mid u_i \neq w_i\big\} \leqslant \mathrm{Card}(A \cup B) \leqslant \mathrm{Card}\,A + \mathrm{Card}\,B = d(u,v) + d(v,w)$$

Question 14 • Nous balayons les deux chaînes en parallèle ; p donne la position courante, c le nombre de différences constatées.

```
let distance u v =
 let (lu,lv) = (string_length u,string_length v) in
 if lu <> lv then failwith "longueur distinctes" else
 let rec aux = function
  | (p,c) when p = lu -> c
  | (p,c) when u.[p] <> v.[p] -> aux(p+1,c+1)
  | (p,c) -> aux(p+1,c)
 in aux (0,0) ;;
```

Question 15 • Γ_n nous donne la liste des sommets par lesquels passer : en effet, la distance qui sépare deux sommets consécutifs est égale à 1 ; de plus, le dernier sommet ($\mathbf{10}^{n-1}$) et le premier ($\mathbf{0}^n$) sont eux aussi à distance 1 l'un de l'autre.

Question 16 • La solution minimale du problème consiste, si $n \geqslant 2$, à déplacer les $n-1$ premiers disques vers la tige 2 ; à déplacer le disque n vers la tige 3 ; et à déplacer une deuxième fois les $n-1$ premiers disques, mais cette fois de la tige 2 vers la tige 3. Une récurrence immédiate montre que le disque à déplacer à la k-ième étape est le numéro du bit qui est modifié, lorsque l'on passe de $\gamma(k-1)$ à $\gamma(k)$ dans le code de Gray réfléchi Γ_n.

Question 17 • L'état de **D** est indifférent : il suffit de faire en sorte qu'à un moment donné, les 3 autres tiges soient dans le même état. Le mot $w =$ **XCXBXCXAXCXBXCX** répond à la question : quel que soit l'état initial, il existe un préfixe (de longueur impaire) de ce mot qui force l'ouverture de la porte. Expliquons le lien avec le code de Gray : considérons le code réfléchi circulaire sur 3 bits (**000**, **001**, **011**, **010**, **110**, **111**, **101**, **100**). Chaque valeur correspond à un état d'une tige (disons : **0** = basse, **1** = haute). Identifions l'appui sur la touche **A** (resp. **B**, **C**) à l'inversion du bit 0 (resp. 1, 2) et supposons, pour fixer les idées, que la tige **D** est en position basse. Quelle que soit la situation (inconnue) depuis laquelle on commence à lire le mot w, on passe par l'état 000, et alors la serrure s'ouvre.

Question 18 • Il n'existe en fait que quatre positions différentes, pour ce problème :

- (1) : un seul verre à l'endroit et les trois autres à l'envers (ou l'inverse) ;
- (2) : deux verres adjacents à l'endroit ;

- (3) : deux verres opposés à l'endroit ;
- (4) : les quatre verres dans la même position.

Il existe trois manœuvres possibles :

- (A) : retourner un seul verre ;
- (B) : retourner deux verres adjacents ;
- (C) : retourner deux verres opposés.

L'idée de base est la suivante : le barman applique d'abord la suite CBC, qui le fait gagner à coup sûr si l'on commence dans l'une des positions (2) ou (3). Sinon, c'est qu'il a commencé dans la position (1) : alors la manœuvre (A) l'amène dans l'une des positions (2), (3) ou (4) ; dans chacun des deux premiers cas, la suite CBC l'amène en (4).

Question 19 • L'automate de la figure 1 répond à la question. Vous vérifierez que la lecture du mot $CBCACBC$ fait passer par l'état 4, quel que soit l'état (autre que 1) dans lequel elle commence. Notez que l'automate a été émondé.

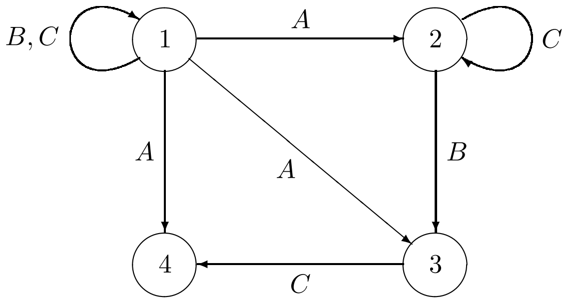

Figure 1: l'automate du barman aveugle

Question 20 • Le «mot magique» du barman aveugle se déduit de celui du professeur Macheprot par effacement des **X**.

Partie 5

▶ Ici apparaît un autre ennui dans le sujet : l'ordre soit-disant «lexicographique» ne l'est pas du tout... Il aurait fallu définir $M_{n+1} = \begin{array}{|c|c|}\hline \mathsf{A}\,M_n & \mathsf{C}\,M_n \\ \hline \mathsf{G}\,M_n & \mathsf{T}\,M_n \\ \hline \end{array}$ où $\mathsf{X}\,M_n$ désigne la puce de même taille que M_n, dans laquelle toutes les séquences ont été préfixées par la lettre X.

Question 21 • Notons B_k la longueur de bord du masque utilisé pour le dépôt de la k-ième base B. Pour l'ordre lexicographique, nous obtenons $A_1 = A_2 = T_1 = T_2 = 4$, et $C_1 = C_2 = G_1 = G_2 = 8$: total 48. Pour l'ordre de Gray, nous obtenons $A_1 = C_1 = G_1 = T_1 = 4$ et $A_2 = C_2 = G_2 = T_2 = 8$; total 48 également.

Question 22 • Le calcul pour le cas «lexicographique» n'a pas lieu d'être.

• Notons $C_{gray}(n)$ la longueur de bord totale des $4n$ masques utilisés pour une puce d'ordre n avec la méthode de Gray. Pour fabriquer une puce d'ordre $n+1$ avec la même méthode, nous aurons besoin de $4n$ masques déduits des précédents par double réflexion : chacun de ces masques a une longueur de bord quadruple de celle du masque d'ordre n dont il est déduit ; ceci parce que le «raccord» le long des médianes n'induit pas de bord. Le coût cumulé est de $4C_{gray}(n)$. Il nous faut également 4 masques couvrant chacun un des quarts de la puce d'ordre $n+1$: leur coût cumulé est 8×2^n. Nous obtenons donc $C_{gray}(n+1) = 4\big(C_{gray}(n) + 2^{n+1}\big)$; comme $C_{gray}(1) = 8$, nous obtenons $C_{gray}(n) = 4^{n+1} - 2^{n+2}$. Remarque : la suite de terme général $C_{gray}(n)/8$ est répertoriée dans l'OEIS (A006516). Détail intéressant, l'article de l'OEIS donne un lien avec : la variante des tours de Hanoï à quatre piquets ; un problème d'énumération de mots sur un alphabet à 4 lettres ; et un problème d'énumération de droites passant par les sommets d'un hypercube ...

Question 23 • Question sans objet (cf. remarque au début de cette partie).

Partie 6

Question 24 Une seule ligne suffit :

```
let ajoute p =  map (fun q -> p::q) ;;
```

Question 25 La formule récursive de l'énoncé se traduit immédiatement ; nous vérifions que l'argument n est positif ou nul.

```
let rec gray = function
 | 0 -> [[]]
 | n when n<0 -> failwith "n positif SVP"
 | n -> let g = gray(n-1) in (ajoute 0 g) @ (ajoute 1 (rev g)) ;;
```

Question 26 Par précaution, nous vérifions que k est positif.

```
let rec puiss2 = function
 | k when k<=0 -> failwith "valeur interdite"
 | 1 -> [1]
 | k -> let r = puiss2(k/2) in (map (prefix * 2) r) @ [1] ;;
```

Question 27 À revoir.

```
let g k =
let rec aux = function
 | (0,_)  -> 0
 | (1,_) -> 1
 | (z,_) when z<0 -> failwith "valeur interdite"
 | (z,[]) -> failwith "erreur de programmation"
 | (z,t::q) when z>=t -> t + aux(t-1-z,q)
 | (z,_::q) -> aux(z,q)
in aux(k,puiss2 k) ;;
```

Question 28 L'approche adoptée ici nous donne un coût en $\lg^2(n)$; on peut obtenir un coût en $2\lg(n)$, avec une récursivité terminale.

```
let rec base2 = function
 | k when k<0 -> failwith "valeur interdite"
 | k when k<2 -> [k]
 | k -> (base2(k/2)) @ [k mod 2] ;;
```

Question 29 La fonction `sumlist` vérifie que les deux listes ont la même longueur.

```
let rec somme2listes = function
 | ([],[]) -> []
 | (a::u,b::v) -> ((a+b) mod 2)::somme2listes(u,v)
 | _ -> failwith "longueurs distinctes" ;;

let g2 = function
 | k when k<0 -> failwith "valeur interdite"
 | k when k<2 -> [k]
 | k -> let u = rev(base2(k)) and v = rev(0::base2(k/2))
   in rev(somme2listes(u,v)) ;;
```

FIN