

# Option Informatique en Spé MP et MP\*

## Morphismes : le corrigé

Cette version, à peu près complète, remplace celle qui a déjà été distribuée, et qui comportait plusieurs erreurs. On notera que la solution de la question 9 ne correspond pas exactement à la situation de l'énoncé : on y suppose définie  $P_0 = \{a\}$ , ce qui simplifie la rédaction.

### Morphismes

**Question 1** •  $f_2 = aba$ ,  $f_3 = abaab$  et  $f_4 = abaababa$ . On note que  $f_2 = f_1 f_0$ ,  $f_3 = f_2 f_1$  et  $f_4 = f_3 f_2$ . Supposons acquise la relation  $f_{n+2} = f_{n+1} f_n$  ; alors  $f_{n+3} = \psi(f_{n+2}) = \psi(f_{n+1} f_n) = \psi(f_{n+1}) \psi(f_n) = f_{n+2} f_{n+1}$  donc la relation est vraie pour tout  $n$ .

**Question 2** •  $|\psi(a)|_a = |ab|_a = 1$ ,  $|\psi(a)|_b = |ab|_b = 1$ ,  $|\psi(b)|_a = |a|_a = 1$  et  $|\psi(b)|_b = |a|_b = 0$ . Donc  $M_\psi = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ .

**Question 3** • Si  $v = \varepsilon$ , alors  $\varphi(v) = \varepsilon$  et  $\mathbf{V}$  est le vecteur nul : les deux membres de l'égalité sont nuls. Sinon, on part de la remarque suivante :  $|v|$  et  $|v|_x$  ne dépendent pas de l'ordre dans lequel on écrit les lettres du mot  $v$ . Alors, pour toute lettre  $x$ , on a :

$$|\varphi(v)|_x = \sum_{y \in A} |\varphi(y)^{|v|_y}|_x = \sum_{y \in A} |v|_y \cdot |\varphi(y)|_x = \sum_{y \in A} |\varphi(y)|_x \cdot |v|_y = \sum_{y \in A} (M_\varphi)_{x,y} \cdot \mathbf{V}_{y,1} = (M_\varphi \cdot \mathbf{V})_x$$

**Question 4** • Une récurrence immédiate donne  $|\varphi^n(v)|_x = ((M_\varphi)^n \cdot \mathbf{V})_x$ . Il suffit ensuite de noter que :

$$\mathbf{H} \cdot (M_\varphi)^n \cdot \mathbf{V} = \sum_{x \in A} \mathbf{H}_{1,x} ((M_\varphi)^n \cdot \mathbf{V})_x = \sum_{x \in A} |\varphi^n(v)|_x = |\varphi^n(v)|$$

**Question 5** • Le produit de deux matrices carrées d'ordre  $p$  peut être calculé en effectuant  $\mathcal{O}(p^3)$  opérations élémentaires ; en utilisant une exponentiation dichotomique, on peut calculer la puissance  $n$ -ième d'une matrice carrée d'ordre  $p$  en effectuant  $\mathcal{O}(\lg n)$  produits de matrices. On peut donc calculer  $(M_\varphi)^n$  au prix de  $\mathcal{O}(|A|^3 \lg n)$  opérations élémentaires. Il reste à effectuer la multiplication à droite par  $\mathbf{V}$ , pour un coût  $\mathcal{O}(|A|^2)$ , puis la multiplication à gauche par  $\mathbf{H}$ , pour un coût  $\mathcal{O}(|A|)$ .

**Question 6** • Le théorème de CAYLEY-HAMILTON affirme que toute matrice carrée d'ordre  $p$  est racine de son polynôme caractéristique, lequel est de degré  $p$  et de coefficient dominant  $(-1)^p$ . Appliquons ceci à la matrice  $M_\varphi$ , carrée d'ordre  $|A|$  ; notons  $P_{M_\varphi}$  son polynôme caractéristique et  $Q_{M_\varphi} = (-1)^{|A|} P_{M_\varphi}$ . Ce polynôme est unitaire, de degré  $|A|$  ; on peut donc écrire  $Q_{M_\varphi} = X^{|A|} - \sum_{1 \leq k \leq |A|} a_k X^{|A|-k}$ . On a  $P_{M_\varphi}(M_\varphi) = 0$ , donc

$Q_{M_\varphi}(M_\varphi) = 0$ , soit  $(M_\varphi)^{|A|} = \sum_{1 \leq k \leq |A|} a_k (M_\varphi)^{|A|-k}$ . Multiplions les deux membres par  $(M_\varphi)^{n-|A|}$ , il vient

$(M_\varphi)^n = \sum_{1 \leq k \leq |A|} a_k (M_\varphi)^{n-k}$ . Multiplions les deux membres par  $\mathbf{H}$  à gauche et par  $\mathbf{V}$  à droite, il vient, avec le

résultat de la question 4 :  $|\varphi^n(v)| = \sum_{1 \leq k \leq |A|} a_k |\varphi^{n-k}(v)|$  soit précisément  $\ell_n = \sum_{1 \leq k \leq |A|} a_k \ell_{n-k}$ .

**Question 7** • Chaque  $a_k$  est une somme (algébrique) de produits d'éléments de  $M_\varphi$  ; comme cette matrice est à coefficients dans  $\mathbb{N}$ , les  $a_k$  sont tous dans  $\mathbb{Z}$ .

### Lettres récurrentes d'un morphisme

**Question 8** • Soient  $a$ ,  $b$  et  $c$  telles que  $a \xrightarrow{\varphi} b$  et  $b \xrightarrow{\varphi} c$ . Il existe un exposant  $n > 0$  tel que  $|\varphi^n(a)|_b \geq 1$  ; ceci revient à dire qu'il existe des mots  $u$  et  $v$  tels que  $\varphi^n(a) = ubv$ . De même, il existe un exposant  $p > 0$  et des mots  $x$  et  $y$  tels que  $\varphi^p(b) = xcy$ . Alors, comme  $\varphi$  est un morphisme :

$$\varphi^{n+p}(a) = \varphi^p(\varphi^n(a)) = \varphi^p(ubv) = \varphi^p(u)\varphi^p(b)\varphi^p(v) = \varphi^p(u)xcy\varphi^p(v)$$

Ceci montre que  $|\varphi^{n+p}(a)|_c \geq 1$  ; comme  $n+p > 0$ , on peut affirmer que  $a \xrightarrow{\varphi} c$ .

**Question 9** • La croissance de la suite  $(P_n)_{n \in \mathbb{N}}$  résulte de sa définition. Si  $P_n \subsetneq P_{n+1}$ , alors  $|P_{n+1}| \geq |P_n| + 1$ ; comme  $|P_0| = 1$ , il existe au plus  $|A| - 1$  indices  $n$  tels que  $P_n \subsetneq P_{n+1}$ . On peut déjà affirmer que la suite stationne, à partir d'un rang que nous noterons  $n_0$ .

• Nous voulons montrer que  $P_n \subsetneq P_{n+1}$  pour  $0 \leq n < n_0$ , et  $n_0 < |A|$ . Il suffit de montrer que, si  $P_n = P_{n+1}$ , alors  $P_{n+1} = P_{n+2}$  : on aura alors  $P_0 \subsetneq P_1 \subsetneq \dots \subsetneq P_{n_0-1} \subsetneq P_{n_0} = P_{n_0+1} = \dots$  avec  $n_0 + 1 \leq |P_{n_0}| \leq |A|$ , donc  $n_0 < |A|$ .

• Seule l'inclusion  $P_{n+2} \subset P_{n+1}$  pose problème. Soit  $x \in P_{n+2}$ ; alors, ou bien  $x \in P_{n+1}$  (et c'est fini), ou bien il existe  $x \in \alpha(\varphi^{n+2}(a))$ . Notant  $\varphi^{n+1}(a) = y_1 y_2 \dots y_q$ , il existe un indice  $j \in \llbracket 1, q \rrbracket$  tel que  $x \in \alpha(\varphi(y_j))$ ; mais  $y_j \in P_{n+1}$ , donc  $y_j \in P_n$  :  $y_j \in \alpha(\varphi^n(a))$ . Par suite,  $x \in P_{n+1}$ .

**Question 10** • On remarque que  $P_n$  est l'ensemble des lettres  $x$  qui apparaissent dans  $\varphi^k(a)$  pour au moins un indice  $k \leq n$ . Donc, si  $a \xrightarrow{\varphi} b$ , il existe un indice  $n > 0$  tel que  $b \in P_n$ ; d'après la question précédente,  $P_n = P_{n_0}$ ; ainsi il existe un indice  $n' \leq n_0$  (donc  $n' < |A|$ ) tel que  $b \in P_{n'}$ .

**Question 11** • Les coefficients de  $(M_\varphi)^k$  sont dans  $\mathbb{N}$ , donc positifs ou nuls. Si  $a \xrightarrow{\varphi} b$ , il existe  $n < |A|$  tel que  $b$  possède au moins une occurrence dans  $\varphi^n(a)$ ; alors  $((M_\varphi)^n)_{b,a} > 0$ ; et par suite  $(T(\varphi, A))_{b,a} =$

$$\left( \sum_{1 \leq k \leq |A|} (M_\varphi)^k \right)_{b,a} = \sum_{1 \leq k \leq |A|} ((M_\varphi)^k)_{b,a} \text{ est strictement positif.}$$

• Réciproquement, si  $(T(\varphi, A))_{b,a} > 0$ , il existe un  $n < |A|$  tel que  $((M_\varphi)^n)_{b,a} > 0$ ; ceci implique que  $b$  possède au moins une occurrence dans  $\varphi^n(a)$  et donc que  $a \xrightarrow{\varphi} b$ .

**Question 12** • Les lettres récurrentes pour  $\varphi$  sont les lettres  $x$  telles que  $(T(\varphi, A))_{x,x} \geq 1$ . Il suffit donc de calculer la matrice  $T(\varphi, A)$ ; ceci revient à effectuer  $|A| - 1$  produits matriciels (coût unitaire :  $|A|^3$  multiplications, et presque autant d'additions) et  $|A| - 1$  additions matricielles (coût unitaire :  $|A|^2$  additions). Le coût total est donc un  $\mathcal{O}(|A|^4)$ .

**Question 13** • Notons  $B = \begin{pmatrix} \mathbf{I} & 0 \\ A & A \end{pmatrix}$ . Alors  $B^2 = \begin{pmatrix} \mathbf{I} & 0 \\ A + A^2 & A^2 \end{pmatrix}$ , puis  $B^3 = \begin{pmatrix} \mathbf{I} & 0 \\ A + A^2 + A^3 & A^3 \end{pmatrix}$ . On vérifie par récurrence que  $B^n = \begin{pmatrix} \mathbf{I} & 0 \\ A + A^2 + \dots + A^n & A^n \end{pmatrix}$  pour tout  $n \in \mathbb{N}$ .

**Question 14** • Avec l'algorithme d'exponentiation rapide de la question 5, on peut élever à la puissance  $|A|$  la matrice  $S = \begin{pmatrix} \mathbf{I} & 0 \\ M_\varphi & M_\varphi \end{pmatrix}$  pour un coût  $\mathcal{O}(|2A|^3 \lg |2A|) = \mathcal{O}(|A|^3 \lg |A|)$ . On obtient ainsi la matrice

$S^{|A|} = \begin{pmatrix} \mathbf{I} & 0 \\ T(\varphi, A) & (M_\varphi)^{|A|} \end{pmatrix}$ . On en déduit la liste des lettres récurrentes en examinant les coefficients diagonaux du bloc  $T(\varphi, A)$ , comme il a été expliqué à la question 12.

## L-systèmes

**Question 15** • Une récurrence immédiate montre que  $\varphi^n(a) = ab^n$ . Donc  $\mathcal{L}_{LS}(\varphi, a) = \{ab^n \mid n \in \mathbb{N}\}$ . Ce langage est rationnel : il est décrit par l'expression rationnelle  $ab^*$ .

**Question 16** •  $\varphi^2(a) = \varphi(ab) = abbb = ab^3$ ;  $\varphi^3(a) = \varphi(ab^3) = ab(bb)^3 = ab^7$ . Supposons  $\varphi^n(a) = ab^{x_n}$ , alors  $\varphi^{n+1}(a) = \varphi(ab^{x_n}) = abb^{2x_n} = ab^{x_{n+1}}$  avec  $x_{n+1} = 2x_n + 1$  soit  $x_{n+1} + 1 = 2(x_n + 1)$ , donc  $x_n + 1 = 2^n(x_0 + 1) = 2^n$  et finalement  $x_n = 2^n - 1$ . Ainsi  $\varphi^n(a) = ab^{2^n - 1}$ .

• Ce langage (que nous noterons  $L$ ) n'est pas rationnel. Pour le prouver, raisonnons par l'absurde : supposons  $L$  rationnel; le lemme de l'étoile affirme l'existence d'une constante  $N$  telle que tout mot  $u$  de  $L$  de longueur au moins égale à  $N$  se décompose en  $u = xyz$ , avec  $|xy| \leq N$ ,  $y \neq \varepsilon$  et  $xy^kz \in L$  quel que soit  $k \in \mathbb{N}$ . Notons  $n$  le naturel défini par  $2^n < N \leq 2^{n+1}$ ; notons  $u = \varphi^{n+1}(a) = ab^{2^{n+1} - 1}$  :  $u \in L$ , et  $|u| = 2^{n+1} \geq N$ . Observons la décomposition de  $u$  dont le lemme de l'étoile affirme l'existence : si  $x = \varepsilon$ , alors  $|y|_a = 1$ , donc  $|xz|_a = 0$  et  $xz$  n'appartient certainement pas à  $L$ . Sinon,  $x = ab^p$ ,  $y = b^q$  avec  $0 < q < 2^{n+1}$  et  $z = b^{2^{n+1} - 1 - q}$ . Alors  $xy^2z = ab^{2^{n+1} - 1 + q}$  n'appartient pas à  $L$  puisque  $2^{n+1} - 1 < 2^{n+1} - 1 + q < 2^{n+1} - 1 + 2^{n+1} = 2^{n+2} - 1$ .

**Question 17** • Il suffit de prendre comme alphabet  $A = \{a, b, c\}$  et de définir  $\varphi$  par  $\varphi(a) = ab$ ,  $\varphi(b) = c$  et  $\varphi(c) = \varepsilon$ . Alors  $\varphi^2(a) = abc$ ; on note que  $\varphi(abc) = abc$ , et donc  $\varphi^n(abc) = abc$  pour tout  $n$ . Conclusion :  $\mathcal{L}_{LS}(\varphi, a) = \{a, ab, abc\}$ .

**Question 18** • Les lettres  $x$  pour lesquelles on n'a pas  $a \xrightarrow{\varphi} x$  n'interviennent pas dans les mots engendrés par le L-système; on peut donc les éliminer de  $A$  sans changer le langage engendré. On notera l'analogie avec l'élimination des états non accessibles d'un automate fini.

**Question 19** • Remarquons déjà que les assertions 1 et 2 ne peuvent être simultanément vraies. Si nous ne sommes pas dans le premier cas, l'ensemble des exposants  $n$  tels que  $\varphi^n(a) = \varepsilon$  est une partie non vide de  $\mathbb{N}^*$  (puisque  $\varphi^0(a) = a \neq \varepsilon$ ). Notons  $n_1$  son plus petit élément, puis  $n_0 = n_1 - 1$ . On a donc  $\varphi^n(a) \neq \varepsilon$  pour tout  $n \leq n_0$ ; par ailleurs, comme  $\varphi(\varepsilon) = \varepsilon$ , on a clairement  $\varphi^n(a) = \varepsilon$  pour tout  $n > n_0$ .

**Question 20** • Pour le sens direct, nous raisonnons par récurrence sur  $|A|$ . Si  $|A| = 1$ , alors  $\mathcal{G}$  est immortel ssi  $\varphi(a) = a^k$  avec  $k \geq 1$ , ce qui revient à dire que  $a$  est récurrente. Supposons le résultat acquis pour tout alphabet de cardinal  $n \geq 1$ . Soient  $A$  un alphabet de cardinal  $n + 1$ ,  $\varphi$  un morphisme de  $A^*$  sur lui-même,  $a \in A$  et  $\mathcal{G} = (\varphi, a)$  un L-système immortel. Si  $a$  est récurrente, la discussion est close; sinon, le mot  $v = \varphi(a)$  n'est pas vide, et la lettre  $a$  n'y apparaît pas. Notons  $Y$  l'alphabet formé des lettres qui apparaissent dans  $v$ :  $|Y| \geq n$ . Il existe au moins une lettre  $y \in Y$  telle que le L-système  $(\varphi, y)$  soit immortel: sinon, notant  $n_y$  le plus petit exposant tel que  $\varphi^{n_y}(y) = \varepsilon$  et  $N = \max_{y \in Y} n_y$ , on aurait  $\varphi^N(v) = \varepsilon$ , donc  $\varphi^{N+1}(a) = \varepsilon$ . Mais alors, de par l'hypothèse de récurrence, l'une des lettres de  $Y$  est récurrente pour  $\varphi$ .

• Réciproquement, soit  $b$  une lettre récurrente; on sait que  $a \xrightarrow{\varphi} b$ . Donc il existe des exposants  $n > 0$  et  $p > 0$  tels que  $|\varphi^n(a)|_b \geq 1$  et  $|\varphi^n(b)|_b \geq 1$ . Alors  $|\varphi^{n+kp}(a)|_b \geq 1$  quel que soit  $k \in \mathbb{N}$ , ce qui montre que  $\mathcal{G}$  est immortel.

**Question 21** • Montrons que la durée de vie maximale d'un L-système mortel est  $|A| - 1$ . Commençons par exhiber un exemple réalisant ce maximum: soient  $A = \{a_1, a_2, \dots, a_r\}$  et  $\varphi$  le morphisme  $\varphi$  défini par  $\varphi(a_k) = a_{k+1}$  pour  $1 \leq k < r$  et  $\varphi(a_r) = \varepsilon$ . Le L-système  $(\varphi, a_1)$  a une durée de vie égale à  $r - 1$ .

• Montrons maintenant que la durée de vie ne peut dépasser  $|A| - 1$ . Procédons par récurrence sur  $|A|$ . Si  $|A| = 1$ , le résultat est clair: le seul L-système mortel sur  $A = \{a\}$  est défini par  $\varphi(a) = \varepsilon$ , et sa durée de vie est nulle. Supposons le résultat acquis pour tout alphabet de cardinal  $n \geq 1$ . Soient  $A$  un alphabet de cardinal  $n + 1$ ,  $\varphi$  un morphisme de  $A^*$  sur lui-même,  $a \in A$  et  $\mathcal{G} = (\varphi, a)$  un L-système mortel.  $a$  ne peut être récurrente, donc l'alphabet  $Y$  formé des lettres qui apparaissent dans  $\varphi(a)$  a un cardinal au plus égal à  $n$ ; pour chaque lettre  $x \in Y$ , le L-système  $(\varphi, x)$  a une durée de vie au plus égale à  $n - 1$  par hypothèse. Donc  $\mathcal{G}$  a une durée de vie au plus égale à  $n$ .

**Question 22** • Soit  $x$  une lettre fortement récurrente pour  $\varphi$  et  $n > 0$  tel que  $|\varphi^n(x)|_x \geq 2$ . D'après le résultat de la question 18, il existe un exposant  $n_0$  tel que  $|\varphi^{n_0}(a)|_x \geq 2$ . Alors, avec une récurrence immédiate,  $|\varphi^{n_0+kn}(a)|_x \geq 2^k$ ; ceci montre qu'il existe dans  $\mathcal{L}_{LS}(\varphi, a)$  des mots de longueur arbitrairement grande, et donc que ce langage est infini.

• Considérons le L-système défini à la question 15:  $\varphi^n(a) = ab^n$ , donc  $\mathcal{L}_{LS}(\varphi, a)$  est infini. Pourtant,  $|\varphi^n(a)|_a = 1$  et  $|\varphi^n(b)|_b = 1$  quel que soit l'exposant  $n$ .

## Vers l'algorithme de Swart

**Question 23** • Cette méthode a un coût exponentiel si la lettre  $a$  est fortement récurrente, ou s'il existe une lettre  $b$  fortement récurrente telle que  $a \xrightarrow{\varphi} b$ .

**Question 24** • La suite  $(|x_k|_{0 \leq k \leq p})$  est croissante, puisque  $|x_k| = |x_{k-1}| + |\varphi^{n-1}(v_k)|$  pour  $1 \leq k \leq p$ . Par hypothèse,  $1 \leq i \leq |\varphi^n(a)|$ , donc  $0 < i \leq |x_p|$ . L'ensemble des indices  $j \in \llbracket 1, p \rrbracket$  tels que  $i \leq |x_j|$  n'est donc pas vide, puisqu'il contient  $p$ . Notons  $k$  son plus petit élément, et montrons que  $x_{k-1} < i$ . Si  $k = 1$ , c'est immédiat puisque  $x_0 = 0$  et  $i \geq 1$ . Sinon,  $k - 1 \geq 1$ , donc  $x_{k-1} < i$  de par la définition de  $k$ . Nous avons établi l'existence d'un indice  $k$  vérifiant  $x_{k-1} < i \leq |x_k|$ .

• Cet indice est unique: si l'on avait à la fois  $x_{k-1} < i \leq |x_k|$  et  $x_{k'-1} < i \leq |x'_k|$ , avec  $k < k'$  pour fixer les idées, on aurait  $k \leq k' - 1$ , donc  $|x_k| \geq |x_{k'-1}|$  puis  $i \leq |x_k| \leq |x_{k'-1}| < i$  ce qui est manifestement contradictoire.

**Question 25** •  $\varphi^n(a) = \varphi^{n-1}(x_{k-1}v_kx_k) = \varphi^{n-1}(x_{k-1})\varphi^{n-1}(v_k)\varphi^{n-1}(x_k)$ . Le choix de  $k$  fait que  $|\varphi^{n-1}(a)| < i \leq |\varphi^{n-1}(a)|$ . Donc la  $i$ -ième lettre de  $\varphi^n(a)$  est la  $i - |x_{k-1}|$ -ième lettre de  $\varphi^{n-1}(v_k)$ . Et ceci se traduit exactement par la formule  $\lambda(\varphi, n, a, i) = \lambda(\varphi, n - 1, v_k, i - |x_{k-1}|)$ .

**Question 26** • Déroulement de l'algorithme: si  $n = 0$ , alors  $i = 1$  donc  $\lambda(\varphi, n, a, i) = a$ . Sinon, on calcule  $v = \varphi(a)$ ; notant  $p = |v|$  et  $v = v_1 \dots v_p$ , on calcule (avec la méthode exposée à la question 5) les  $|\varphi^{n-1}(v_k)|$  pour  $1 \leq k \leq p$ . Par sommation, on en déduit les  $|x_k|$  pour  $0 \leq k \leq p$ ; on détermine alors le plus petit indice  $k$  tel que  $|x_k| \geq i$ . Nous sommes alors ramenés à la détermination de  $\lambda(\varphi, n - 1, v_k, i - |x_{k-1}|)$ .

**Question 27** • Comme  $p \leq d_\varphi$ , la première phase du calcul a un coût total  $\mathcal{O}(|A|^3 \lg(n-1)d_\varphi) = \mathcal{O}(|A|^3 d_\varphi \lg n)$ . La deuxième phase a un coût  $\mathcal{O}(d_\varphi)$ , dominé par l'autre coût. D'où la majoration demandée.

**Question 28** • Il suffit d'effectuer une sommation.

**Question 29** • Comme la suite des  $|x_k|$  est croissante, on peut effectuer une recherche dichotomique. Détaillons : si  $|v| = 1$ , c'est fini. Sinon, découpons  $v$  en deux parts égales (à 1 près) :  $v = ww'$ . Calculons  $q = \lfloor \varphi^{n-1}(w) \rfloor$ . Si  $i \leq q$ , on remplace  $v$  par  $w$  ; sinon, on remplace  $v$  par  $w'$  et  $i$  par  $i - q$  ; on recommence jusqu'à ce que le mot dans lequel on cherche la position soit de longueur 1. Ceci permet de remplacer le facteur  $d_\varphi$  par  $\lg(d_\varphi)$ .

## Programmation en Caml

*I conclude that there are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.*

Charles Antony Richard Hoare — 1980 Turing Award Lecture

**Question 30** • On construit l'intervalle  $\llbracket 0, n - 1 \rrbracket$ , où  $n$  est la longueur de la chaîne ; on remplace ensuite chaque indice par la lettre ayant ce rang.

```
let list_of_string u =
  map (fun i -> u.[i]) (intervalle 0 (string_length u - 1)) ;;
```

**Question 31** • On transforme la liste en chaîne ; on remplace chaque caractère par son image par  $\varphi$ , en utilisant `map` et `assoc` ; on concatène les membres de la liste de chaînes ainsi obtenue.

```
let applique_morphisme phi u =
  concat ( map (fun x -> assoc x phi) (list_of_string u) ) ;;
```

**Question 32** • On utilise  $\varphi^n(u) = \varphi^{n-1}(\varphi(u))$ .

```
let rec itere_morphisme phi u = fonction
  | 0 -> u
  | n -> itere_morphisme phi (applique_morphisme phi u) (n-1) ;;
```

**Question 33** • Immédiat, avec la fonction `itere_morphisme`. Ne pas oublier que l'indexation des chaînes de caractères commence à 0.

```
let lambda phi n a i =
  (itere_morphisme phi a n).[i-1] ;;
```

**Question 34** • On vérifie que les dimensions sont compatibles ; ensuite, il y a trois boucles imbriquées.

```
let produit_matrices a b =
  let n = vect_length a and p = vect_length a.(0)
  and p' = vect_length b and q = vect_length b.(0) in
  if p <> p' then failwith "dimensions incompatibles" else
  let c = make_matrix n q 0 in
  for i = 0 to n-1 do
  for j = 0 to q-1 do
  for k = 0 to p-1 do
    c.(i).(j) <- c.(i).(j) + a.(i).(k)*b.(k).(j)
  done; done; done; c ;;
```

L'utilisation de `c.(i).(j)` comme «accumulateur» n'est pas une excellente idée, car la sommation des  $a_{i,k}b_{k,j}$  s'effectue dans un emplacement mémoire, et non dans un registre. Deux solutions : utiliser une référence (hérésie) ; ou écrire une fonction qui calcule le produit scalaire de la  $i$ -ième ligne de  $a$  par la  $j$ -ième colonne de  $b$ , ce qui est facile avec `it_list` et `map`. La programmation n'est pas totalement fonctionnelle, mais les éléments de la matrice ne sont modifiés qu'une fois, après l'initialisation.

```

let produit_matrices a b =
  let n = vect_length a and p = vect_length a.(0)
  and p' = vect_length b and q = vect_length b.(0) in
  if p<>p' then failwith "dimensions incompatibles" else
  let proscal i j = it_list (prefix +) 0
    (map (fun k -> a.(i).(k)*b.(k).(j)) (intervalle 0 (p-1))) in
  let c = make_matrix n q 0 in
  for i = 0 to n-1 do
    for j = 0 to q-1 do c.(i).(j) <- proscal i j
  done; done; c ;;

```

**Question 35** • La fonction `identite` construit la matrice identité d'ordre  $n$ . L'utilisation d'un opérateur infixe `***` simplifie l'écriture.

```

let prefix *** = produit_matrices ;;

let identite nl =
  let r = make_matrix nl nl 0 in
  for i = 0 to nl-1 do r.(i).(i) <- 1 done;
  r ;;

let exponentielle_matrice m n =
  let nl = vect_length m and nc = vect_length m.(0) in
  if nl <> nc then failwith "matrice rectangulaire"
  else
  let rec aux mat accu = function
    | 0 -> accu
    | n when n mod 2 = 1 -> let mat' = mat *** mat in
      let accu' = accu *** mat in
      aux mat' accu' (n/2)
    | n -> let mat' = mat *** mat in aux mat' accu (n/2)
  in aux m (identite nl) n ;;

```

**Question 36** • La fonction `compte_lettres`, de signature `'a -> 'a list -> int`, calcule le nombre d'occurrences d'un objet dans une liste. On construit une nouvelle représentation de  $\varphi$ , en remplaçant la chaîne de caractères  $\varphi(x)$  par la liste des caractères de cette chaîne; dans un deuxième temps, pour accélérer l'accès à chaque couple, on transforme la liste décrivant  $\varphi$  en un vecteur.

```

let compte_lettres x u = list_length (filtre (prefix = x) u) ;;

let matrice_of_morphisme phi =
  let n = list_length phi
  and v = vect_of_list (map (fun (x,y) -> (x,list_of_string y)) phi) in
  let m = make_matrix n n 0 in
  for i = 0 to n-1 do
    for j = 0 to n-1 do
      m.(i).(j) <- compte_lettres (fst v.(i)) (snd v.(j))
    done; done; m ;;

```

**Question 37** • On commence par la partie pénible: construire la matrice  $S$  de la question 14. C'est du bête travail de copie: un `blit_matrix` serait bienvenu!

```

let construire_s phi =
  let a = matrice_of_morphisme phi in
  let n = vect_length a in
  let s = make_matrix (2*n) (2*n) 0 in
  for i = 0 to n-1 do
    s.(i).(i) <- 1 ;

```

```

for j = 0 to n-1 do
  s.(i+n).(j) <- a.(i).(j) ;
  s.(i+n).(j+n) <- a.(i).(j)
done
done; s ;;

```

Et maintenant, on calcule  $S$ ; on l'élève à la puissance  $p = |A|$ ; on construit la liste des couples  $(x_i, i)$  où  $0 \leq i < p$  et les  $x_i$  sont les lettres de  $A$ ; on filtre cette liste pour ne garder que les lettres récurrentes: ce sont celles dont l'indice  $i$  vérifie  $S_{i+p,i} > 0$ ; enfin, on projette les couples pour ne garder que les lettres. Simple, non?

```

let lettres_récurrentes phi =
  let s = construire_s phi and p = list_length phi in
  let s' = exponentielle_matrice s p in
  let w = combine (map fst phi, intervalle 0 (p-1)) in
  map fst (filtre (fun (_,i) -> s'.(i+p).(i) > 0) w) ;;

```

**Question 38** • Les fonctions `row_of_vect` et `col_of_vect` transforment un vecteur en une matrice à une ligne ou une colonne respectivement. Ceci permet d'utiliser le produit matriciel défini plus haut dans la fonction `longueur`; cette dernière calcule  $|\varphi^n(u)|$ .

```

let row_of_vect v = [|v|] ;;

let col_of_vect v = map_vect row_of_vect v ;;

let longueur phi n u =
  let nl = list_length phi in
  let m = matrice_of_morphisme phi in
  let m' = exponentielle_matrice m n in
  let v = row_of_vect (make_vect nl 1) in
  let cl z = compte_lettres (fst z) u in
  let u = col_of_vect (vect_of_list (map cl phi)) in
  (v *** m' *** u).(0).(0) ;;

```

`skip` calcule, à partir de la donnée de  $i$  et des  $x_j$ , l'indice  $k$  défini à la question 24, et la «nouvelle» valeur de  $i$ , à savoir  $i - \sum_{1 \leq j < k} x_j$ . `lambda` calcule  $\lambda(\varphi, n, u, i)$ .

```

let rec skip = fonction
  | (k,j,t:::q) when j>t -> skip (k+1,j-t,q)
  | (k,j,t::_) -> (k,j)
  | _ -> failwith "i est trop grand" ;;

let rec lambda phi a i = fonction
  | 0 -> a.[i-1]
  | n -> let v = applique_morphisme phi a in
  let p = string_length v in
  let interv = intervalle 0 (p-1) in
  let g = fun k -> longueur phi (n-1) [v.[k]] in
  let x = map g interv in
  let (k,i') = skip (0,i,x) in
  lambda phi (string_of_char v.[k]) i' (n-1) ;;

```

FIN