Exercice 1. *Ens ULM-Lyon-Cachan MP* 2003*

1) Soit A un alphabet fini contenant au moins deux lettres distinctes. Soit L le langage des palindromes sur A. L est-il régulier ?

2) Proposer un algorithme déterministe prenant en entrée un automate quelconque $\mathcal{A}$ et déterminant si $\mathcal{A}$ reconnaît un palindrome.

Exercice 2. *Ens Cachan MP* 2003*

1) Soit $n \in \mathbb{N}^*$. On considère l'automate $\mathcal{A} = (I = \{0\}, F = \{0\}, Q = \{0, 1, \ldots, n-1\}, \delta)$ où δ est définie par les triplets suivants (état,lettre,état) :

$$\{(i, 0, i),\ i \geqslant 1\} \cup \{(i, 0, 0),\ i \geqslant 1\} \cup \{(i, 1, i+1),\ i \leqslant n-2\} \cup \{(n-1, 1, 0)\}.$$

Montrer que l'automate déterministe obtenu par la méthode du cours comporte 2^n états.

2) On considère le langage L_n des mots sur $A = \{a, b\}$ tels que la n-ème lettre avant la fin soit un a. Montrer que tout automate déterministe reconnaissant L_n comporte au moins 2^n états.

Exercice 3. *Ens ULM-Lyon-Cachan MP* 2003*

On appelle *langage local sur un alphabet* A un langage de la forme $L = (XA^* \cap A^*Y) \setminus (A^*VA^*)$ où $X \subset A$, $Y \subset A$, $V \subset A^2$.

1) Prouver qu'un tel langage est reconnaissable.

2) Donner un exemple non trivial de langage reconnaissable non local.
La réponse « Aucun langage local ne contient ε donc $L = A^*$ est un contre-exemple » a été refusée (trop facile). L'examinatrice suggère $L = (ab)^*(ac)^*$ sur l'alphabet $\{a, b, c\}$.

3) Prouver qu'un langage local sur un alphabet à n lettres peut être reconnu par un atuomate déterministe complet à $n+2$ états.

4) Montrer que si $R \subset A^*$ est reconnaissable alors il existe un alphabet B un langage $L \subset B^*$ local et une application $\varphi : B \longrightarrow A$ tel que $\varphi^*(L) = R$ avec $\varphi^* : \begin{cases} B^* & \longrightarrow & A^* \\ u = b_1 \ldots b_n & \longmapsto & \varphi^*(u) = \varphi(b_1)\ldots\varphi(b_n). \end{cases}$

Exercice 4. *Ens ULM-Lyon-Cachan MP* 2003*

Soit $\mathcal{G}$ un graphe donné par un ensemble S de sommets et un ensemble E d'arêtes orientées.

1) Écrire un algorithme prenant en entrée le graphe $\mathcal{G}$, un sommet v_0, et qui renvoie la liste des sommets atteignables en partant de v_0 (on choisira une représentation des arêtes, puis le programme prendra E, S, v_0 en entrée). On démontrera la terminaison et la correction de cet algorithme, et on calculera sa complexité.

2) En déduire un algorithme répondant à la question : $\mathcal{G}$ a-t-il un cycle ?

3) Trouver un algorithme répondant à la question précédente en $\mathrm{O}(\operatorname{card} E + \operatorname{card} S)$. On introduira une fonction $\sigma : S \longrightarrow \mathbb{N}$ telle que : $\forall\ s_1, s_2 \in S,\ \overrightarrow{s_1 s_2} \in E \Longrightarrow \sigma(s_1) < \sigma(s_2)$.

Exercice 5. *Ens Ulm-Lyon MP* 2003*

Soit $\mathcal{G}$ un graphe connexe non orienté, S l'ensemble des sommets et A l'ensemble des arêtes. Le degré $d(s)$ d'un sommet s est le nombre d'arrêtes ayant s comme sommet. Sur chaque sommet, on place $c(s)$ pions. On obtient alors une configuration $c : S \longrightarrow \mathbb{N}$.

Règle du jeu : si un sommet s possède au moins $d(s)$ pions, on distribue un pion à chacun de ses voisins, obtenant ainsi une nouvelle configuration c'. On note $c \xrightarrow{s} c'$ et on dit que s a été joué.

1) Écrire un algorithme qui joue tant qu'il peut.

2) Montrer que si une partie est infinie, alors tous les sommets sont joués une infinité de fois.

3) Montrer que si une partie est finie, alors l'un des sommets n'est pas joué.

4) Donner une condition nécessaire sur le nombre de pions pour qu'il existe une partie finie.

5) Montrer que si une partie est finie alors toutes les autres parties possibles à partir de la même configuration initiale sont aussi finies et aboutissent à la même configuration finale.

Exercice 6. *Ens Ulm-Lyon MP* 2003*

Soit A un tableau de n entiers relatifs. On veut trouver le sous-tableau connexe de A dont la somme des éléments est maximale, c'est-à-dire $i \leqslant j$ tels que $\sum_{k=i}^{j} A[k]$ est maximal.

1) Donner un algorithme simple résolvant le problème.

2) Évaluer sa complexité.

3) Donner un nouvel algorithme utilisant la méthode « diviser pour régner ».

4) Quelle est sa complexité ?

Solutions

Exercice 1.

1) Non, il contredit le lemme de l'étoile.

2) Former l'automate $\mathcal{A}'$ ayant pour états les couples (q, q') où q et q' sont des états de $\mathcal{A}$ et pour transitions $(q, q') \xrightarrow{a} (r, r')$ avec $q \xrightarrow{a} q'$ et $r' \xrightarrow{a} r$ dans $\mathcal{A}$. On prend comme états initiaux de $\mathcal{A}'$ les couples (i, f) où i est l'état initial et f un état final de $\mathcal{A}$. Soit u un mot quelconque. On a un chemin $(i, f) \xrightarrow{u} (q, q')$ dans $\mathcal{A}'$ si et seulement si $i \xrightarrow{u} q$ et $q' \xrightarrow{\tilde{u}} f$ sont des chemins dans $\mathcal{A}$. Ainsi, le langage $L(\mathcal{A})$ contient un palindrome de longueur paire $u\tilde{u}$ si et seulement s'il existe un chemin dans $\mathcal{A}'$ de la forme $(i, f) \xrightarrow{*} (q, q)$ et $L(\mathcal{A})$ contient un palindrome de longueur impaire $ua\tilde{u}$ si et seulement s'il existe dans $\mathcal{A}'$ un chemin de la forme $(i, f) \xrightarrow{*} (q, q')$ avec $q \xrightarrow{a} q'$ dans $\mathcal{A}$.

Exercice 2.

1) Soit X un ensemble d'états accessibles construit lors de l'algorithme de déterminisation. Alors $X \cup \{0\}$ et l'image de X par la permutation circulaire $0 \longrightarrow 1 \longrightarrow \ldots \longrightarrow n-1 \longrightarrow 0$ seront aussi construits lors de cet algorithme. Et tout sous-ensemble de $\{0, \ldots, n-1\}$ peut être obtenu à partir de $\varnothing$ par un nombre fini de ces transformations (récurrence sur le cardinal).

Remarque : le programme officiel n'impose pas de méthode de déterminisation, il pourrait y avoir d'autres algorithmes que celui des sous-ensembles.

Remarque : un mot u est reconnu par $\mathcal{A}$ si et seulement s'il est constitué de 0 et de 1 et s'il se termine par un 0 ou si le nombre d'occurences de la lettre 1 est nul ou au moins égal à n. On construit facilement un automate déterministe à $n+1$ états reconnaissant ce langage.

2) Si $u \in (a+b)^n$ alors la connaissance du résiduel $u^{-1}L_n = \{v \in (a+b)^* \text{ tq } uv \in L_n\}$ permet de déterminer u : la k-ème lettre de u est un a si et seulement si $ua^{n-k} \in L_n$. Si un automate $\mathcal{A}$ déterministe reconnaît L_n alors $\mathcal{A}$ a au moins autant d'états qu'il y a de mots distints dans $(a+b)^n$, soit 2^n.

Exercice 3.

1) Cours.

2) Si $L = (XA^* \cap A^*Y) \setminus (A^*VA^*)$ alors $X = \{a\}$ (seule première lettre possible), $Y = \{b, c\}$ (dernières lettres possibles) et $V = \{aa, bb, cc, bc, cb\}$ (doublets exclus). Et ça ne marche pas, $acab \in (XA^* \cap A^*Y) \setminus (A^*VA^*)$.

3) On prend un état initial I, un état rebut R, et un état x pour chaque lettre de A. Puis on place les transitions $x \xrightarrow{y} y$ pour toutes lettres x, y telles que $xy \notin V$, les transitions $I \xrightarrow{x} x$ pour toutes les lettres de X, on fait pointer les transitions manquantes vers l'état rebut. Les états acceptants sont ceux étiquetés par une lettre de Y.

4) Soit $\mathcal{A}$ un automate déterministe reconnaissant R et Q l'ensemble des états de $\mathcal{A}$. On prend $B = Q \times A \times Q$ et $\varphi(q, a, q') = a$. Un mot de B^* représente un chemin dans l'automate $\mathcal{A}$ si et seulement s'il est de la forme :

$$u = (q_1, a_1, q_2)(q_2, a_2, q_3)\ldots(q_{n-1}a_{n-1}q_n), \qquad q_i \in Q,\ a_i \in A$$

et dans ce cas $\varphi^*(u) = a_1 \ldots a_n$ est le mot étiquetant ce chemin. Soit L l'ensemble des mots u tels que $\varphi^*(u) \in R$. Comme φ^* est surjective, on a $\varphi^*(L) = R$. De plus L est local : X est l'ensemble des transitions autorisées par $\mathcal{A}$ partant de l'état initial de $\mathcal{A}$, Y est l'ensemble des transitions aboutissant à un état final de $\mathcal{A}$ et V est l'ensemble des doublets inacceptables (transitions non définies dans $\mathcal{A}$ ou état intermétiaire incorrect).

Exercice 4.

1) On suppose que les sommets sont étiquetés par les entiers de $[\![0, s-1]\!]$ où $s = \mathrm{card}(S)$. On choisit de représenter E par une matrice booléenne $M \in \mathcal{M}_s(\{\texttt{true},\texttt{false}\})$ telle que $M(i,j)$ vaut `true` si et seulement s'il existe une arête du sommet i vers le sommet j. La liste résultat sera représentée par un vecteur V tel que $V(i)$ vaut `true` si et seulement s'il existe un chemin dans $\mathcal{G}$ menant de v_0 à i. Algorithme :

$V \longleftarrow [\text{false}, \ldots, \text{false}]$ *; P* $\longleftarrow$ *pile vide ; empiler* v_0.
Tant que P est non vide :
dépiler le sommet i de P ; $V(i) \longleftarrow$ `true`
pour $j = 0 \ldots s-1$ *: si* $M(i,j)$ *et non*$(V(j))$ *alors empiler j ; finpour*
fintantque

Terminaison et correction : il y a terminaison car chaque sommet est empilé au plus une fois sur P. La liste V obtenue ne contient que des sommets accessibles depuis v_0 par récurrence sur le nombre d'itérations de la boucle *Tant que*. Tout sommet i accessible depuis v_0 est effectivement placé dans V par récurrence sur la longueur d'un chemin menant de v_0 à i.

Complexité : chaque sommet dépilé donne lieu à $\mathrm{O}(s)$ opérations, donc la complexité est $\mathrm{O}(sN(v_0))$ où $N(v_0)$ est le nombre de sommets accessibles depuis v_0. Dans le pire des cas, on a une complexité $\mathrm{O}(s^2)$.

2) Dans l'algorithme précédent, il y a un cycle passant par v_0 si et seulement si l'on dépile un sommet i tel que $M(i, v_0)$. En faisant varier v_0 on obtient un détecteur de cycle de complexité $\mathrm{O}(s^3)$.

3) Une fonction σ telle que décrite dans l'énoncé est appelée : *ordre topologique sur* $\mathcal{G}$. On ordonne topologiquement un graphe orienté sans cycle en choisissant un sommet sans prédecesseur que l'on place en tête de la liste résultat ; on supprime ce sommet du graphe et on itère. Si à un moment on ne trouve pas de sommet sans prédecesseur alors il y a un cycle.

Mais en fait il n'y a pas besoin de faire tout ça : l'algorithme donné en **1)** tourne en $\mathrm{O}(\mathrm{card}\, E + \mathrm{card}\, S)$ si l'on représente les arrêtes par des *listes d'adjacence* (on note pour chaque sommet la liste de ses successeurs). En effet, chaque arrête du graphe est examinée au plus une fois lors de la recherche des sommets accessibles à partir de v_0.

Exercice 5.

2) Sinon, l'ensemble E des sommets joués une infinité de fois s'appauvrit d'au moins un pion à chaque fois qu'on joue un sommet de E ayant un voisin hors de E, et il existe un tel sommet si $E \neq S$ car $\mathcal{G}$ est connexe.

3) On note pour $s \in S$, $n(s)$ le nombre de fois où s a été joué. On suppose $n = \min\{n(s),\ s \in S\} > 0$ et on considère le sommet $s_0 \in S$ qui a atteint le premier son compte $n(s_0)$ au cours du jeu. Tous les voisins de s_0 ont été joués au moins une fois après que s_0 se soit mis à bouder, donc à la fin de la partie s_0 dispose d'un capital suffisant pour redonner un pion à chacun de ses voisins, contradiction.

4) Le nombre total de pions, $\sum_{s\in S} c(s)$, est invariant au cours d'une partie. Lorsqu'on arrive à la fin d'une partie, chaque sommet a strictement moins de pions que son degré, d'où : $\sum_{s\in S} (d(s) - c(s)) \geqslant \mathrm{card}(S)$.

5) On démontre par récurrence sur $n \in \mathbb{N}$ la proposition suivante : *Soit c une configuration et P une partie commençant avec c finie de longueur n. Alors toute partie Q commençant avec c est finie et aboutit à la même configuration c′ que P.*

Le cas $n = 0$ est trivial. Dans le cas où $n \geqslant 1$, soit s_0 le premier sommet joué au cours de P et $s_1, \ldots, s_k$ les sommets joués au cours de Q avant s_0 (on doit jouer s_0 dans Q sinon Q serait finie et dans la configuration finale de Q le sommet s_0 aurait un capital de pions supérieur ou égal à $c(s_0) \geqslant d(s_0)$, impossible). Au lieu de jouer s_k puis s_0, on peut jouer s_0 puis s_k car les deux sommets ont suffisament de pions pour cela et le compte de pions pour chaque sommet après ces deux coups est le même. De proche en proche, on peut ramener s_0 en tête des coups joués, ce qui donne une nouvelle partie R qui aboutit à la même configuration que Q après les $k+1$ premiers coups. En particulier R est finie ou infinie comme Q, et le cas échéant les configurations finales de Q et R sont égales. Mais R commence par le même premier coup que P, donc l'hypothèse de récurrence à l'ordre $n-1$ s'applique aux parties P', R' de P et R qui suivent ce premier coup.

Exercice 6.

1) Calculer toute les sommes et retenir les indices fournissant la plus grande.

2) $O(n^3)$ si l'on recalcule chaque somme à partir de zéro. On peut faire $O(n^2)$ en calculant de proche en proche toutes les sommes qui commencent au même indice i.

3) On partage le tableau en deux moitiés, on cherche récursivement dans chaque moitié : la plus grande sous-somme, la plus grande sous-somme commençant au début, la plus grande sous-somme se terminant à la fin et la somme totale de la moitié considérée. Ces huit informations permettent de déterminer en temps constant les quatre renseignements associés pour le tableau complet.

4) linéaire.