Groupes C/S et M et I

ÉPREUVE D'INFORMATIQUE

(Sujet commun ENS : ULM et LYON)

DURÉE : 4 heures

L'usage de la calculatrice n'est pas autorisé.

Important

Les différentes parties sont largement indépendantes. **Il est indispensable de lire attentivement le Préambule avant d'aborder l'une quelconque des parties du problème.**

On attachera une grande importance à la clarté et à la précision des réponses fournies.

Les algorithmes que l'on demande de décrire seront d'abord expliqués (ou décrits à l'aide de "pseudo-code" en décrivant les étapes) avant d'être présentés à l'aide de lignes de programme dans un langage laissé au choix du candidat, dans les cas (et **seulement** dans les cas) où il apparaîtra utile de compléter cette description.

Préambule

Le problème a pour objet d'étudier des structures permettant de "représenter" ou de "caractériser" des formes arborescentes telles que celle représentée sur la figure 1.

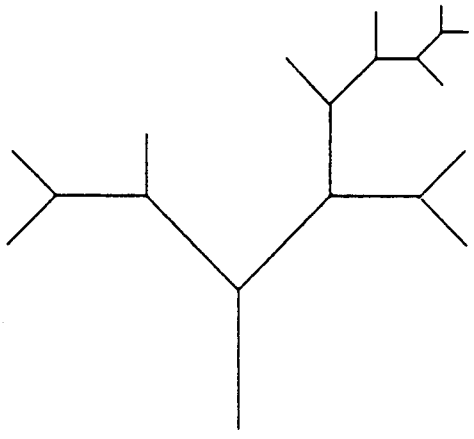

Figure 1: Une "forme arborescente"

On montre que certaines structures (matrices dans la première partie, mots dans la troisième partie) permettent de "coder" des propriétés intéressantes de ces formes et qu'elles peuvent être utilisées pour en produire des dessins.

On supposera dans tout le problème que les formes sont des formes planes (dessins dans le plan de formes arborescentes). Les techniques présentées peuvent être étendues à l'espace tridimensionnel. Elles forment la base de techniques effectivement utilisées pour construire des "modèles" informatiques de plantes ou d'arbres (de la nature) en synthèse d'images.

La première partie du problème décrit une technique permettant d'associer à une forme arborescente une matrice. Celle-ci est telle que certaines propriétés visuelles de la forme (être touffu ou effilé, par exemple – voir figures 2 et 3) se traduisent par des propriétés de la matrice. On dispose alors, grâce à un processus "inverse", qui associe une forme arborescente à une matrice, d'un procédé pour produire (synthétiser) des formes ayant certaines particularités, en particulier du point de vue de leur aspect visuel.

Dans la deuxième partie, on considère l'arbre associé de manière classique à une expression arithmétique. On montre que l'un des paramètres calculé lors de la construction de la matrice associée définie dans la partie 1 est relié à un paramètre qui s'introduit naturellement lors du calcul de la valeur de cette expression (le nombre de "registres" utilisé par la machine pour l'évaluation de l'expression).

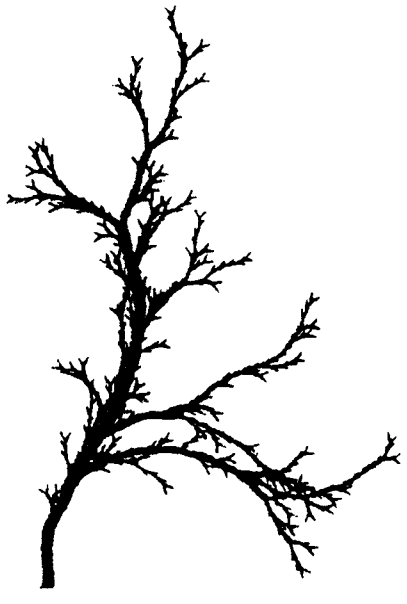

Figure 2: Une représentation de forme arborescente effilée

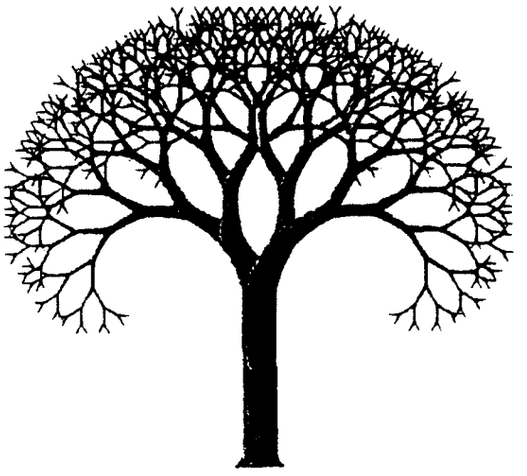

Figure 3: Une représentation de forme arborescente touffue

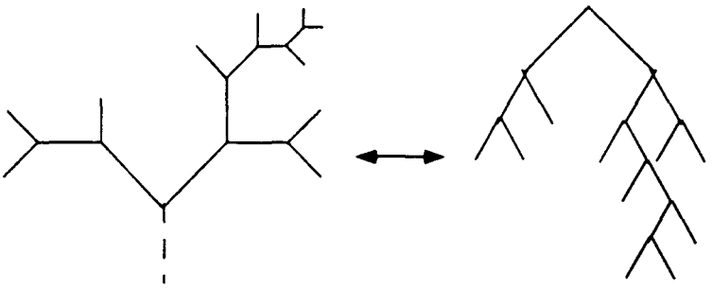

Figure 4: L'arbre binaire sous-jacent à une forme arborescente

La troisième partie du problème étudie une représentation de forme arborescente sous forme de mot d'un certain langage.

La forme d'un "arbre" dépend à la fois de sa structure topologique (type d'embranchements, façon dont ils sont organisés) et de propriétés géométriques telles qu'épaisseur ou longueur de chacun des segments de branches, angles entre branches, etc. On s'intéresse, dans un premier temps, seulement à la structure topologique des arbres.

On considère la **forme arborescente** représentée sur la figure 1. On notera que l'orientation et la longueur des différentes **arêtes** qui la composent peuvent être modifiées sans que soit modifiée la structure topologique de la forme. On notera également que l'on a représenté le "tronc" de l'arbre par une arête particulière. Sur la figure 4, on illustre le fait qu'après suppression de cette arête tronc, symétrie par rapport à une horizontale (et redimensionnement et positionnement des arêtes), on obtient un **arbre binaire**, au sens où on l'entend habituellement en Informatique : tous les nœuds de l'arbre ont zéro, un ou deux fils (on dit qu'ils sont de **degré** 0, de degré 1, ou de degré 2 ; les **feuilles** sont les nœuds de degré 0 ; les autres nœuds sont appelés **nœuds internes** et sont de degré 1 ou 2 pour un arbre binaire "général"). L'arbre construit comme indiqué sur la figure 4 est tel que tous ses nœuds internes sont de degré 2 (à cause des hypothèses faites sur les formes arborescentes considérées) ; on dit que c'est un **arbre binaire strict**.

On supposera dans les première et deuxième parties que tous les embranchements sont "d'ordre 2", en d'autres termes que la structure topologique sous-jacente est un **arbre binaire strict**.

Dans la troisième partie du problème, on n'impose plus cette condition sur les degrés des nœuds et l'on étudie des codages de formes arborescentes plus générales, qui seront précisées en début de troisième partie.

On définit la **taille** d'un *arbre* comme étant le nombre de ses nœuds internes. On définit la **taille** d'une *forme arborescente* comme étant égale à la taille de l'arbre qui lui est associé par le procédé illustré sur la figure 4.

On définit la **hauteur** d'un *arbre* comme étant le nombre maximum de nœuds internes rencontrés lorsque l'on suit (en "descendant" dans l'arbre binaire) un chemin menant de sa racine jusqu'à une de ses feuilles (en incluant la racine et la feuille dans le décompte). On définit la **hauteur** d'une *forme arborescente* comme étant égale à la hauteur de l'arbre qui lui est associé par le procédé illustré sur la figure 4.

Ainsi, la forme arborescente et l'arbre considérés sur la figure 4 ont pour taille 9 et pour hauteur 6.

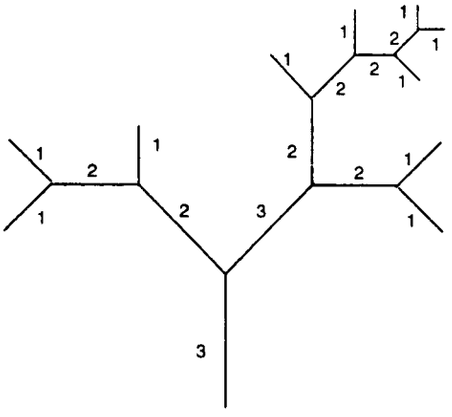

Figure 5: Une forme arborescente et les épaisseurs de ses arêtes

Première Partie

Dans le but de "caractériser" des formes arborescentes, telle que celle représentée en figure 1, du point de vue de leur structure de branchement, on introduit un étiquetage des arêtes de l'arbre. Par abus de langage, nous utiliserons indifféremment dans la suite les termes forme arborescente et arbre ; selon le contexte, un arbre sera donc soit une forme arborescente telle que celle représentée sur la figure 1 - qui possède une arête particulière codant le tronc - , soit un arbre au sens habituel, tel que celui représenté sur la partie droite de la figure 4.

L'étiquette que l'on associe à une *arête* sera appelée son **épaisseur**. Les épaisseurs des arêtes d'un arbre sont obtenues en attribuant l'épaisseur 1 à toutes les arêtes issues des feuilles, puis en calculant récursivement les épaisseurs des autres arêtes en utilisant les règles données sur la figure 6 (on calcule l'épaisseur de l'arête "verticale" représentée sur cette figure en fonction des épaisseurs des deux autres arêtes). Le résultat obtenu pour l'arbre de la figure 1 a été représenté sur la figure 5.

En d'autres termes :

$$\begin{cases} k = \max(i,j) & \text{si} \quad i \neq j \\ k = i + 1 & \text{si} \quad i = j \end{cases}$$

On appelle **épaisseur** d'une *forme arborescente*, l'épaisseur de son arête issue de la racine. Par abus de langage, l'**épaisseur** de l'*arbre* associé à cette forme est définie comme étant égale à l'épaisseur de la forme. On notera $ep(t)$, l'épaisseur de l'arbre t. Ce paramètre vaut 3 dans la cas de la forme arborescente représentée sur la figure 5.

Etant donné un nœud interne x de l'arbre, tel que celui représenté sur la figure 6, on définit la **biépaisseur** de ce *nœud* comme étant le couple (i,j) (avec $i < j$), ou le couple $(k-1, k-1)$, portés sur la figure. On notera que, si k est donné, la biépaisseur de x peut prendre k valeurs distinctes : $(1,k)$, $(2,k)$, ..., $(k-1,k)$ et $(k-1,k-1)$.

Les notions d'épaisseur et de biépaisseur ayant été définies, on associe à tout arbre t, une matrice $R(t)$ appelée **matrice de ramification** de t. Cette matrice est une matrice carrée ayant $ep(t)$ lignes et colonnes. Si l'on note $N_t(i,j)$ le nombre de nœuds de l'arbre t de biépaisseur (i,j) et $N_t(k)$ le nombre d'arêtes de t d'épaisseur k, l'élément (k,i) de la matrice $R(t)$ est défini par $R_{1,1} = 1$, $R_{1,i} = 0$ si $j > 1$, et, pour $k \geq 2$:

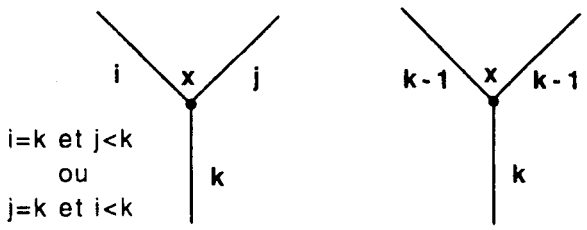

Figure 6: Biépaisseur d'un nœud

$$\begin{cases} R_{k,i} = N_t(i,k)/N_t(k) & \text{si} \quad i < k \\ R_{k,k} = N_t(k-1,k-1)/N_t(k) & \\ R_{k,i} = 0 & \text{si} \quad i > k \end{cases}$$

Chaque ligne de la matrice de ramification correspond à une épaisseur donnée k et donne la "distribution" du nombre de nœuds de biépaisseur (i,k) selon les valeurs de i : l'élément $R_{k,i}$ a pour valeur la proportion de nœuds de l'arbre ayant pour biépaisseur (k,i) pour $i = 1,\ldots,k-1$, et $R_{k,k}$ a pour valeur la proportion de nœuds de l'arbre ayant pour biépaisseur $(k-1,k-1)$.

Question 1

Calculer la matrice de ramification de l'arbre représenté sur la figure 1.

Question 2

Que vaut la somme des éléments d'une ligne d'une matrice de ramification ? Justifier.

Question 3

Un arbre **parfait** est un arbre binaire dont tous les niveaux sont "remplis" jusqu'à une **profondeur** (nombre d'arêtes traversées en suivant un chemin partant de la racine et allant vers une feuille) égale à la hauteur de l'arbre. La figure 7 donne un exemple de tel arbre.

Montrer qu'un arbre binaire strict est un arbre parfait si et seulement si tous les chemins menant de sa racine à une feuille quelconque ont pour longueur (comptée en nombre d'arêtes traversées) la hauteur de l'arbre.

Donner, en fonction de h, la matrice de ramification d'un arbre parfait de hauteur h.

Question 4

Donner, en fonction de h, la matrice de ramification d'un arbre **peigne** de hauteur h. Un arbre peigne est un arbre binaire tel que le sous-arbre gauche de tous ses nœuds internes soit réduit à une feuille. La figure 8 donne un exemple de tel arbre.

Question 5

On observe que les matrices de ramification obtenues en réponse aux questions 3 et 4 sont très différentes. Quelle intuition tire-t-on de ces exemples quant au lien (ou aux liens) existant entre la forme d'un arbre

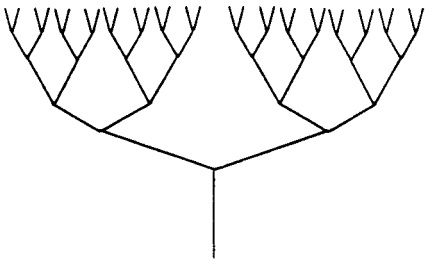

Figure 7: Une forme arborescente correspondant à un arbre parfait

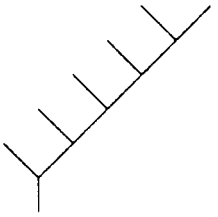

Figure 8: Une forme arborescente correspondant à un arbre peigne

et la répartition des valeurs de coefficients de la matrice de ramification associée ? Donner quelques arguments pouvant servir de support à l'intuition. On ne demande pas ici de preuve.

Question 6

Donner, en fonction de n, une borne inférieure $\lambda(n)$ sur le paramètre épaisseur d'un arbre de taille n. Combien d'arbres de taille n ont-ils pour épaisseur $\lambda(n)$?

Question 7

Discuter les valeurs possibles du paramètre $n(h)$, nombre de nœuds internes d'un arbre ayant une épaisseur h donnée. Justifier.

Question 8

Comment une forme arborescente peut-elle être stockée en machine ? De quels paramètres liés à cette forme dépend l'encombrement mémoire ? Comment en dépend-il ? Les réponses à ces questions sont-elles fonction du langage de programmation utilisé ?

Question 9

Proposer un algorithme de calcul de l'épaisseur d'une forme arborescente. Evaluer la complexité de cet algorithme en fonction de la taille n de la forme arborescente.

Question 10

Pour engendrer des arbres binaires stricts ayant tous une même taille donnée n mais des formes différentes, on peut partir d'un arbre réduit à un nœud interne et deux feuilles et procéder de manière récursive conformément à la méthode suivante : à chaque étape, on choisit l'une des feuilles de l'arbre déjà construit que l'on remplace par un nœud interne et les deux feuilles qui lui sont "attachées" ; on s'arrête quand on a construit un arbre de taille n.

On suppose que l'on dispose d'une fonction `r1` qui renvoie, à chacun de ses appels, un nombre compris entre 0 et 1 (r1 dépend d'un paramètre, non explicité ici, qui est recalculé à chaque appel ; son but est d'assurer - ceci pouvant être précisé - une bonne répartition sur l'intervalle $]0,1[$ des valeurs renvoyées lors de différents appels de la fonction).

Proposer un algorithme utilisant cette fonction et construisant un arbre à l'aide de la méthode exposée ci-dessus. Evaluer la complexité de cet algorithme en fonction de n.

Question 11

Pour synthétiser une forme arborescente, on peut aussi partir d'une matrice de ramification et se servir des éléments de cette matrice pour construire pas à pas un arbre comme suit.

On se donne une matrice triangulaire inférieure M, telle que la somme des éléments de chacune de ses lignes soit égale à 1. Soit s le nombre de ses lignes. Ce sera l'épaisseur de l'arbre engendré.

On part d'un arbre t_1 réduit à une seule arête étiquetée s. Cette arête correspond au tronc de la forme arborescente. On construit ensuite récursivement une suite d'arbres binaires étiquetés : l'arbre t_{n+1}, de taille $n+1$, est obtenu à partir de l'arbre t_n en choisissant une arête terminale de t_n (c'est-à-dire ayant une feuille pour l'une de ses extrémités ; on considère que c'est le cas de l'arête de t_1) et en la prolongeant par deux arêtes filles. Supposons que l'arête terminale considérée soit étiquetée k (avec $k \neq 1$). On utilise alors une fonction dépendant de k, $r2_k$ (construite "dans le même esprit" que la fonction r1 introduite à la question 10), qui renvoie, à chacun de ses appels, un entier compris entre 1 et k. Si cette fonction renvoie une valeur $i < k$, on étiquette les arêtes filles avec les valeurs k et i ; si la fonction renvoie la valeur k, les arêtes filles sont étiquetées $k-1$ et $k-1$.

Donner les grandes lignes d'un algorithme permettant de construire un arbre en utilisant la méthode exposée ci-dessus. Est-on assuré qu'un tel algorithme se termine ?

Question 12

La technique présentée à la question 11 est effectivement utilisée pour synthétiser des formes arborescentes. Elle est intéressante si l'on utilise une famille de fonctions telle que pour k donné, la fonction $r2_k$ dépende de la k-ième ligne de la matrice M : étant donnés l'entier k et les valeurs $M_{k,1}, M_{k,2}, \ldots, M_{k,k}$ (dont la somme est égale à 1), la fonction $r2_k$ renvoie un entier i compris entre 1 et k, calculé de telle sorte que sur un très grand nombre d'appels de la fonction la proportion de ceux pour lesquels la fonction renvoie la valeur i est approximativement égale à $M_{k,i}$. On admettra qu'une telle famille de fonctions $r2_k$ existe.

Que peut-on espérer quant aux propriétés des formes arborescentes engendrées par cette technique ? (On ne demande pas de preuves des conjectures proposées en réponse à cette question.)

Deuxième Partie

Les modèles usuels de machines comportent une unité de calcul opérant sur un petit nombre de registres ; les opérandes, généralement issus de calculs antérieurs, sont pris dans une mémoire de capacité plus grande, et chargés dans des **registres** sur lesquels l'unité de calcul opère. Un modèle simplifié d'un évaluateur d'expressions arithmétiques est décrit par la figure 9.

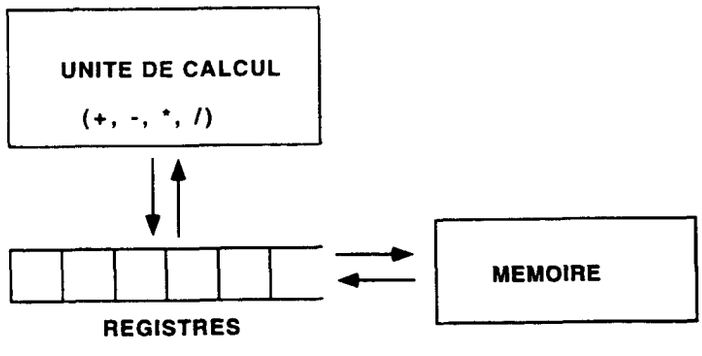

Figure 9: Modèle de machine

Les instructions de base de la machine sont des instructions

- de *calcul* (de la forme $R_i \leftarrow R_j$ op R_k où op est l'un quelconque des opérateurs binaires d'addition, de soustraction, de multiplication ou de division ($+, -, *$ ou $/$),
- de *chargement* dans un registre (de la forme $R_i \leftarrow MEM[j]$),
- de *rangement* en mémoire (de la forme $MEM[j] \leftarrow R_i$).

Un **programme-machine** est une suite d'instructions séparées par des points-virgules.

Le nombre de registres de calcul utilisés par une telle machine pour calculer une expression dépend de l'ordre d'évaluation des sous-expressions. Si l'on considère, par exemple, l'expression $U = a*(b+c)$, elle peut être calculée par évaluation de gauche à droite selon le programme

$$R_0 \leftarrow a \ ; \ R_1 \leftarrow b \ ; \ R_2 \leftarrow c \ ; \ R_1 \leftarrow R_1 + R_2 \ ; \ R_1 \leftarrow R_0 * R_1.$$

Cela demande 3 registres, tandis que l'évaluation de droite à gauche n'en demande que 2, puisqu'elle peut être réalisée par le programme :

$$R_0 \leftarrow b \ ; \ R_1 \leftarrow c \ ; \ R_0 \leftarrow R_0 + R_1 \ ; \ R_1 \leftarrow a \ ; \ R_0 \leftarrow R_1 * R_0.$$

Question 13

Ecrire le programme-machine correspondant à l'évaluation de gauche à droite de l'expression

$$V = a + (b - c) * (d - e).$$

Combien ce programme utilise-t-il de registres ?

Question 14

Ecrire le programme-machine correspondant à l'évaluation de droite à gauche de l'expression V. Combien ce programme utilise-t-il de registres ?

Question 15

On considère la stratégie consistant à calculer en priorité les expressions demandant pour leur évaluation le plus grand nombre de registres.

Ainsi, supposons que l'on veuille calculer $Expr_1$ op $Expr_2$ où $Expr_1$ et $Expr_2$ sont deux expressions arithmétiques et op un opérateur binaire (addition, soustraction, multlipication ou division). Lorsque l'évaluation de $Expr_1$ requiert un nombre de registres supérieur ou égal au nombre de registres requis par l'évaluation de $Expr_2$, on évalue $Expr_1$ puis $Expr_2$; enfin on calcule $Expr_1$ op $Expr_2$. Dans le cas contraire, on évalue $Expr_2$ puis $Expr_1$; enfin, on calcule $Expr_1$ op $Expr_2$.

Ecrire le programme-machine correspondant à l' évaluation de l'expression

$$W = a + (b - c) * (d - e)/(f + g)$$

en utilisant la stratégie exposée ci-dessus. Combien ce programme utilise-t-il de registres ?

[On démontre, ce que l'on admettra ici, que cette stratégie est **optimale** : elle utilise un nombre minimal de registres].

Question 16

Donner une représentation sous forme d'arbre de l'expression arithmétique W.

Question 17

A partir de l'arbre représentant une expression arithmétique, on construit, en "oubliant" les étiquettes portées par ses nœuds (variables ou constantes aux feuilles, opérateurs binaires aux nœuds internes), un arbre binaire au sens classique. En opérant une symétrie autour d'une droite horizontale, puis en ajoutant une arête "tronc" issue de la racine de cet arbre binaire, on obtient une forme arborescente.

Illustrer cette construction sur l'exemple de l'expression W.

Montrer que le nombre de registres utilisés lors de l'évaluation de l'expression arithmétique à l'aide de la stratégie optimale exposée à la question 15 est relié (on précisera comment) à l'épaisseur de la forme arborescente associée ci-dessus à l'expression arithmétique (la notion d'épaisseur d'une forme arborescente a été introduite au début de la première partie du problème).

Troisième Partie

On n'impose plus dans la suite du problème la condition vérifiée jusqu'ici que tous les nœuds internes des arbres sont de degré exactement égal à 2.

On supposera **désormais** que les nœuds internes peuvent être de degré 1 ou 2, ce qui permet de considérer par exemple la forme arborescente représentée sur la partie gauche de la figure 10.

Les arbres seront dans la suite codés par des mots d'un langage approprié. Le principe du codage est de faire correspondre un symbole à chacun des éléments constituant la forme (symboles que l'on va retrouver dans le mot) et de "marquer" par des symboles spécifiques l'apparition d'une "branche" : le codage d'une branche sera encadré par des crochets : [et]. Pour illustrer les correspondances entre les divers éléments dans le codage d'un arbre par un mot, on a associé sur la figure 10(ii) une étiquette particulière à chacun des éléments constitutifs de l'arbre. Le codage associé est le mot :

$$w = ab[cd]ef[g[h]i]j[klm]no.$$

En fait, sauf dans le cas où l'on souhaiterait marquer de manière spécifique certains éléments de l'arbre, on se contente d'associer à tous les éléments constitutifs de l'arbre le même symbole a. C'est ce qu'on fera dans **toute** la suite du problème.

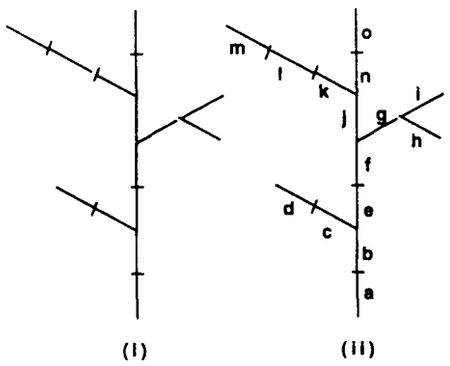

Figure 10: Une forme arborescente sans contrainte de degré 2 sur ses nœuds

Pour formaliser ces notions, on appelle Σ l'alphabet réduit au symbole a : $\Sigma = \{a\}$, et on considère deux autres symboles, [et], appelés **délimiteurs de branches**, et un autre symbole #, appelé **marqueur de branche**.

On note Σ^* (resp. Σ^+) l'ensemble de tous les mots (resp. l'ensemble de tous les mots non vides) construits sur l'alphabet Σ, Λ le mot vide, $|w|$ la longueur du mot w, w_1w_2 la concaténation des mots w_1 et w_2. On considère aussi $\Sigma_E = \Sigma \bigcup \{[,]\}$ et $\Sigma_\# = \Sigma \bigcup \{\#\}$, et on définit comme ci-dessus Σ_E^*, Σ_E^+, $\Sigma_\#^*$ et $\Sigma_\#^+$.

On dit qu'un mot construit sur Σ_E est **bien emboîté** si et seulement s'il peut être construit en appliquant un nombre fini de fois les règles :

- tout mot u de Σ^* est bien emboîté,
- si u et v sont deux mots de Σ_E^* bien emboîtés, alors $[uv]$ et uv sont aussi bien emboîtés.

Question 18

Donner le codage dans Σ_E^* de la forme arborescente représentée sur la figure 10(i).

Question 19

Une branche d'une forme arborescente correspond, dans le codage de l'arbre sous forme d'un mot u de Σ_E^*, à un sous-mot de u, qui est de la forme $[w]$. Donner une condition nécessaire et suffisante sur w pour que le sous-mot $[w]$ de u soit bien l'image, par le codage, d'une branche de la forme arborescente. On dira alors, pour abus de langage, que le *mot* $[w]$ est une **branche**.

Question 20

Montrer qu'une branche $[w] \in \Sigma_E^*$ peut être décomposée de manière unique sous la forme :

$$[w] = [x_1[\alpha_1]x_2[\alpha_2]\dots x_n[\alpha_n]x_{n+1}]$$

où les sous-mots $x_1, x_2, \dots, x_{n+1}$ sont des éléments de Σ^* et les sous-mots $[\alpha_1], [\alpha_2], \dots, [\alpha_n]$ sont des branches.

Les mots $x_1x_2\dots x_nx_{n+1}$ et $x_1\#x_2\#\dots\#x_n\#x_{n+1}$ sont appelés respectivement l'**axe** et l'**axe marqué** de $[w]$. Les mots $[\alpha_1], [\alpha_2], \dots, [\alpha_n]$ sont appelés **branches latérales** de w.

Question 21

Caractériser les mots de $\Sigma_\#^*$ qui peuvent être obtenus comme axes marqués de branches de codages de formes arborescentes.

Construire un automate permettant de reconnaître le langage formé par ces mots.

Question 22

Caractériser les mots de Σ_E^* qui correspondent à des formes arborescentes qui, du point de vue de leur seule structure de branchement, peuvent être considérés comme des arbres parfaits (voir Partie 1, Question 3).

Proposer un algorithme permettant de tester si un mot est bien le codage d'une telle forme.

Question 23

Caractériser les mots de Σ_E^* qui correspondent à des formes arborescentes qui, du point de vue de leur seule structure de branchement, peuvent être considérés comme des arbres peignes (voir Partie 1, Question 4).

Construire un automate permettant de reconnaître le langage formé par ces mots.

Question 24

Proposer un algorithme permettant de tester si un mot de Σ_E^* est bien le codage d'une forme arborescente.

Question 25

On souhaite produire un dessin d'arbre compatible avec la structure de branchement codée par un mot donné de Σ_E^*. On suppose qu'on dispose de deux fonctions, la fonction `tracer` qui, étant donné les coordonnées d'un point P du plan, trace un segment de droite entre le "point courant" (que l'on a initialisé en début de tracé) et le point P, et la fonction `aller` qui, étant donné les coordonnées d'un point P, positionne le point courant en P.

Proposer un algorithme qui trace une forme arborescente admettant pour codage un mot donné u. On supposera que u est bien le codage d'une forme arborescente. On ne se préoccupera pas d'"optimiser" la clarté du dessin obtenu. En particulier, on ne traitera pas des problèmes causés par les intersections entre traits sur le dessin.

www.tunisie-etudes.info

Ce document a été téléchargé depuis
www.tunisie-etudes.info

Des documents gratuits, devoirs, examens, cours, exercices, corrigés... Ainsi que toute une rubrique pour vous aider à trouver un emploi sans oublier les avis de concours en direct

Notre page Twitter :
http://www.twitter.com/TunisieEtudes
Notre page FaceBook :
http://www.facebook.com/TunisieEtudes



Merci d'avoir choisi www.tunisie-etudes.info
Bonne lecture et bon travail