INFORMATIQUE

DURÉE : 5 heures

Les deux parties sont obligatoires ; il est suggéré aux candidats de consacrer trois heures au problème 1 et deux heures au problème 2.

DÉFINITIONS DE BASE

1. **Mots.**

Un *alphabet* est un ensemble fini de symboles, appelés *lettres,* tous distincts. On appelle *mot,* sur un alphabet A, une suite finie d'éléments de A :

$$(a_1, a_2, \ldots, a_n),\ a_i \in \mathrm{A}.$$

La *concaténation* est une opération binaire définie par :

$$(a_1, a_2, \ldots, a_n) \cdot (b_1, b_2, \ldots, b_p) = (a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_p).$$

La concaténation est associative et permet d'écrire un mot sous la forme suivante :

$$a_1 a_2 \ldots a_n$$

au lieu de $(a_1) \cdot (a_2) \ldots (a_n)$ en identifiant une lettre $a \in \mathrm{A}$ avec la suite (a) et en omettant le symbole « · » lorsque celui-ci n'est pas utile à la compréhension.

La suite vide, appelée *mot vide,* est l'élément neutre pour la concaténation et est notée 1. On a donc, pour tout mot w :

$$1 \cdot w = w \cdot 1 = w.$$

L'ensemble des mots sur A est noté A^*. Un *langage* est un ensemble de mots. On notera A^+ l'ensemble des mots non vides sur A : $\mathrm{A}^+ = \mathrm{A}^* - \{1\}$. La *longueur* d'un mot $w = a_1 a_2 \ldots a_n$ est le nombre n de lettres composant w et est notée $|w|$; la longueur du mot vide est $0 : |1| = 0$. On a alors $|u \cdot v| = |u| + |v|$. On notera A^n l'ensemble des mots sur A de longueur égale à n, $\mathrm{A}^0 = \{1\}$.

Un mot $f \in \mathrm{A}^+$ est un *facteur* d'un mot $w \in \mathrm{A}^*$ s'il existe deux mots $f_1, f_2 \in \mathrm{A}^*$ tels que $w = f_1 \cdot f \cdot f_2$; f est un *facteur gauche* (respectivement un *facteur droit*) de w si $f_1 = 1$ (respectivement si $f_2 = 1$). Un *morphisme* h de A^* dans B^* est une application telle que $h(u \cdot v) = h(u) \cdot h(v)$ pour tous mots u et v ; h est, en fait, complètement défini par sa restriction de A vers B^* car pour tout mot $w = a_1 a_2 \ldots a_n$ de A^*, on a :

$$h(a_1 a_2 \ldots a_n) = h(a_1)\, h(a_2) \ldots h(a_n).$$

2. **Ensembles.**

Le cardinal d'un ensemble E est noté $|\mathrm{E}|$. $\mathcal{N}$ désigne l'ensemble des entiers positifs et $\mathcal{Z}$ est l'ensemble des entiers positifs et négatifs. On ajoute à $\mathcal{N}$ (et à $\mathcal{Z}$) un élément noté ∞ tel que pour tout entier n, l'addition et l'ordre habituel sur $\mathcal{Z}$ s'étendent de la manière suivante :

$$\forall n \in \mathcal{Z}, \qquad n + \infty = \infty + n = \infty + \infty = \infty \qquad \text{et} \qquad n < \infty.$$

On étend encore l'addition et l'ordre habituel sur $(\mathcal{Z} \cup \{\infty\})^p$, avec p un entier supérieur ou égal à 1 : la somme de deux vecteurs u et v de $(\mathcal{Z} \cup \{\infty\})^p$ et la relation d'ordre $\leqslant$ sur $(\mathcal{Z} \cup \{\infty\})^p$ sont définies par :

$w = u + v$ si et seulement si pour tout $i = 1, \ldots, p$ $\quad w(i) = u(i) + v(i)$

$u \leqslant v$ si et seulement si pour tout $i = 1, \ldots, p$ $\quad u(i) \leqslant v(i)$.

Pour simplifier l'écriture, on notera $\mathcal{N}_\infty^p$ l'ensemble $(\mathcal{N} \cup \{\infty\})^p$. Pour un vecteur $v \in \mathcal{N}_\infty^p$, on note $v(i)$ sa i-ème composante. La limite d'une suite croissante v_n de $\mathcal{N}_\infty^p$ est définie comme suit : $v = \lim v_n$ avec :

si $\max(v_n(i) / n \in \mathcal{N}) = k_i \in \mathcal{N}$ **alors** $v(i) = k_i$;

sinon $v(i) = \infty$.

Soit $(\mathrm{F}, \leqslant)$ un ensemble ordonné ; *Max (F)* est l'ensemble des éléments maximaux de $(\mathrm{F}, \leqslant)$, c'est-à-dire :

$$\mathrm{Max}\ (\mathrm{F}) = \{f \in \mathrm{F} \ /\ \text{si } (f \leqslant g \text{ et } g \in \mathrm{F}) \text{ alors } f = g\}.$$

Un *graphe étiqueté* G est donné par un triplet $\mathrm{G} = (\mathrm{N}, \mathrm{E}, \mathrm{A})$ où N est l'ensemble des nœuds, E est un alphabet dont les lettres étiquettent les arcs et A est l'ensemble des arcs étiquetés, c'est-à-dire une partie de $\mathrm{N} \times \mathrm{E} \times \mathrm{N}$. Un *arc étiqueté* sera noté (n, e, n'), ou encore $n \text{—}e\text{→} n'$ (ne pas confondre cette notation avec celle de la relation d'accessibilité). Un *circuit* dans un graphe étiqueté G est une suite d'arcs étiquetés où le premier nœud coïncide avec le dernier :

$$n_1 \text{—}e_1\text{→} n_2 \text{—}e_2\text{→} n_3 \text{—}e_3\text{→} \ldots \text{→} n_1.$$

Tournez la page S.V.P.

PROBLÈME 1

On se propose d'étudier quelques propriétés des Systèmes d'Addition de Vecteurs.

Un *Système d'Addition de Vecteurs* (qu'on notera SAV) $S = (V, T, v_0)$ de dimension $(p, n) \in (\mathcal{N} - \{0\})^2$ est la donnée d'un ensemble fini $V = \{v_1, ..., v_n\}$ de n vecteurs de $\mathcal{Z}^p$, d'un alphabet $T = \{t_1, ..., t_n\}$ à n lettres et d'un vecteur $v_0 \in \mathcal{N}^p$.

Soient deux points u et v de $\mathcal{N}_\omega^p$, on dit que v est *S-accessible* (ou *accessible* lorsqu'il n'y a pas de confusion à propos du SAV S considéré) *depuis* u lorsqu'il existe une suite de vecteurs (avec répétitions possibles) $w_1, ..., w_k$ dans V telle que (1) et (2) soient satisfaits.

$$v = u + w_1 + ... + w_k \qquad (1)$$

$$\text{pour } i = 1 \text{ à } n, \quad \text{on a} \quad u + w_1 + ... + w_i \in \mathcal{N}_\omega^p \qquad (2)$$

Cette relation d'accessibilité se note $u\text{--}x\text{-->}\ v$ où $x = t_{i_1} ... t_{i_k}$ avec $w_1 = v_{i_1}, ..., w_k = v_{i_k}$.
(***Remarque :*** le mot $x = t_{i_1} ... t_{i_k}$ indique la suite de vecteurs v_i ajoutés à u pour obtenir v.)

Soit un SAV $S = (V, T, v_0)$. L'ensemble $\mathcal{A}(S)$ des vecteurs accessibles de S est appelé l'*ensemble d'accessibilité* de S et est défini par :

$$\mathcal{A}(S) = \{ v \in \mathcal{N}^p \,/\, \exists x \in T^* ;\ v_0\text{--}x\text{-->}\ v \}.$$

On définit aussi la *i-ème projection* de l'ensemble $\mathcal{A}(S)$ par :

$$\mathcal{A}_i(S) = \{ v(i) \,/\, v \in \mathcal{A}(S) \}.$$

Le *langage* de S est l'ensemble :

$$\mathcal{L}(S) = \{ x \in T^* \,/\, \exists v \in \mathcal{A}(S);\ v_0\text{--}x\text{-->}\ v \}.$$

On notre $\mathcal{SAV}(p, n)$ l'ensemble des SAV de dimension (p, n), $\mathcal{SAV}$ l'ensemble des SAV de dimensions quelconques et $\mathcal{L}_{\mathcal{SAV}}$ l'ensemble des langages de SAV.

On se propose de trouver trois algorithmes résolvant les trois problèmes suivants :

- P1 : étant donné un SAV S, $\mathcal{A}(S)$ est-il fini ?
- P2 : étant donnés un SAV S et un entier $i \in \{1, ..., p\}$, $\mathcal{A}_i(S)$ est-il fini ?
- P3 : étant donné un SAV S, $\mathcal{L}(S)$ est-il fini ?

1. Ensembles et graphes de couverture.

1.1. Montrer que $\mathcal{SAV}$ est dénombrable.

1.2. Montrer que toute suite infinie de vecteurs de $\mathcal{N}_\omega^p$ contient une sous-suite croissante.

1.3. Montrer que pour tout SAV $S = (V, T, v_0)$ on a :

$$\forall t_i \in T,\ \forall v, v' \in \mathcal{N}_\omega^p,\ (v < v' \Rightarrow (v\text{--}t_i\text{-->}\ w \Rightarrow (\exists w' \in \mathcal{N}_\omega^p \,/\, v'\text{--}t_i\text{-->}\ w' \text{ et } w < w'))).$$

En déduire :

$$\forall x \in T^*,\ \forall v, v' \in \mathcal{N}_\omega^p,\ (v < v' \Rightarrow (v\text{--}x\text{-->}\ w \Rightarrow (\exists w' \in \mathcal{N}_\omega^p \,/\, v'\text{--}x\text{-->}\ w' \text{ et } w < w'))).$$

On dira que les SAV sont *monotones.*

Définition.

Un *ensemble de couverture* d'un SAV $S = (V, T, v_0)$ est un sous-ensemble E de $\mathcal{N}_\omega^p$ tel que les conditions (1) et (2) soient satisfaites :

1. $\forall v \in \mathcal{A}(S),\ \exists e \in E,\ v \leqslant e.$
2. $\forall e \in E,\ \forall k \in \mathcal{N},\ \exists x_k \in T^*,\ \exists v_k, v_{k+1} \in \mathcal{A}(S),\ v_k \leqslant v_{k+1},\ v_k\text{--}x_k\text{-->}\ v_{k+1}$ et $\lim v_k = e.$

Notation.

On note $\mathcal{EC}(S)$ l'ensemble des ensembles de couverture de S.

1.4. Donner un ensemble de couverture E_1 ainsi que $\text{Max}(E_1)$ du SAV $S_1 = (V_1, T_1, v_{0_1})$ avec :

$V_1 = \{v_1 = (-2, 1),\ v_2 = (1, -1),\ v_3 = (2, -1),\ v_4 = (-1, 1)\}$;
$T_1 = \{t_1, t_2, t_3, t_4\}$;
$v_{0_1} = (1, 1)$.

1.5. Soient E_1 et E_2 deux ensembles de couverture d'un SAV S.

1.5.1. Montrer que Max (E_1) est fini.

1.5.2. Montrer que Max (E_1) = Max (E_2).

Définition.

Pour tout SAV S, on appelle *ensemble de couverture minimal* l'ensemble ECM (S) = $\cap_{E \in \mathscr{E}(S)} E$.

1.6. Montrer que ECM (S) est fini.

1.7. Montrer que : $\mathscr{A}$ (S) fini $\Leftrightarrow$ ECM (S) est inclus dans $\mathscr{N}^p$.

1.8. Montrer que : $\mathscr{A}_i$ (S) infini $\Leftrightarrow$ $\exists \nu \in$ ECM (S) tel que $\nu(i) = \infty$.

Définition.

Le *graphe de couverture minimal* d'un SAV S = (V, T, ν_0) est le graphe étiqueté GCM (S) = (N, E, A) tel que :

N = ECM (S) ;

E = T ;

A = $\{(\nu, t_i, \nu') / \nu, \nu' \in$ ECM (S) et $\nu + \nu_i = \nu'\}$.

(***Remarque :*** entre deux nœuds, il peut ne pas y avoir d'arc.)

1.9. Construire le graphe de couverture minimal de S_1.

1.10. Montrer que : $\mathscr{L}$ (S) infini $\Leftrightarrow$ il existe un circuit dans GCM (S).

1.11. Donner le principe d'un algorithme qui calcule ECM (S).

1.12. Montrer que votre algorithme se termine en un nombre fini d'étapes et qu'il calcule effectivement ECM (S).

Indications : vous pourrez utiliser le lemme de König que vous démontrerez aussi.

Lemme de König : tout arbre infini (= ayant un nombre infini de nœuds) de degré fini (= chaque nœud a un nombre fini de successeurs directs) contient une branche infinie.

2. Langages de SAV.

On note $\mathscr{A}^k$ (S) = $\{\nu \in \mathscr{A}$ (S) / $\exists x \in T^k$; $\nu_0 \text{--}x\text{-->} \nu\}$.

Nous allons majorer $|\mathscr{A}^k(S)|$ et en déduire une propriété sur les langages de SAV.

2.1. Soit B (k, n) = $|\{(k_1, \ldots, k_n) / \sum_{i=1,\ldots,n} k_i = k\}|$. Montrer que $|\mathscr{A}^k(S)| \leqslant B(k, n)$.

2.2. Calculer B (k, n).

2.3. Montrer que pour $k \geqslant n$, B (k, n) est majoré par $2^n k^n$.

2.4. Calculer N tel que pour $k \geqslant$ N, on ait : B (k, n) $< n^k$.

2.5. En déduire que L = $\{xx / x \in T^*\} \notin \mathscr{L}_{\mathscr{GAV}}$.

2.6. Déterminer une classe $\mathscr{L}$ de langages telle que $\mathscr{L} \cap \mathscr{L}_{\mathscr{GAV}} = \varnothing$.

Tournez la page S.V.P.

PROBLÈME 2

Le but du problème est d'étudier un algorithme de génération aléatoire d'arbres binaires.

Un *arbre binaire* est soit un sommet unique (*la racine* dans ce cas), soit un sommet (*la racine*) relié par une arête à un arbre binaire (*le sous-arbre gauche*) et par une autre arête à un autre arbre binaire (*le sous-arbre droit*).

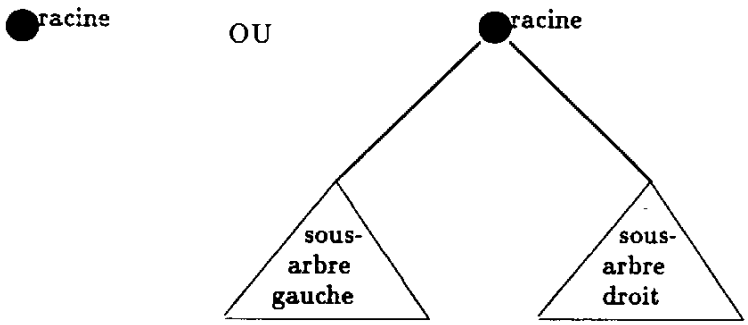
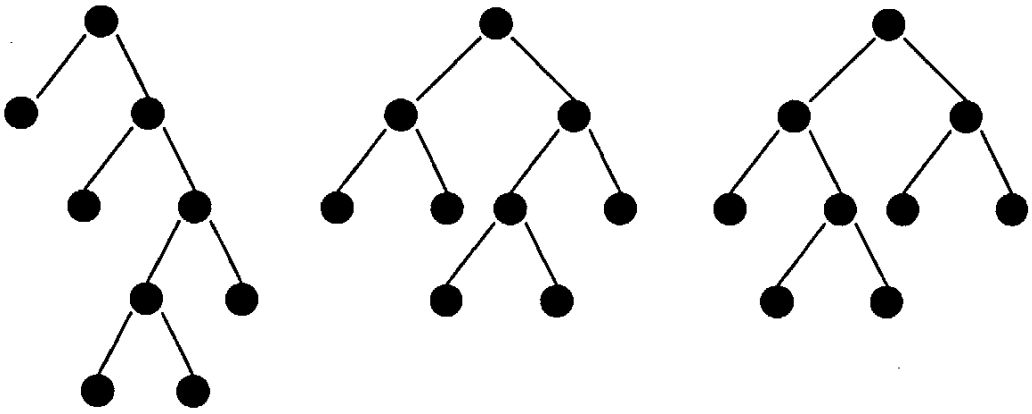


Schéma de la définition récursive des arbres binaires



Trois exemples d'arbres binaires à neuf sommets

Le *langage de Dyck* (noté D) sur l'alphabet $A = \{a, b\}$ est défini comme étant l'ensemble des mots f tels que soit $f = 1$, soit f peut se décomposer en $f = a f_1 b f_2$ avec $f_1 \in D$ et $f_2 \in D$.

En machine, les arbres binaires seront représentés par un pointeur sur le sommet racine. Chaque sommet sera une structure contenant trois pointeurs sur des sommets : le premier pointant sur la racine du sous-arbre gauche, le second pointant sur la racine du sous-arbre droit et le troisième pointant sur le sommet père. Pour la racine, le pointeur sur le sommet père est NIL. Si l'arbre est uniquement constitué de sa racine, les deux pointeurs de la structure correspondante seront NIL (c'est le cas pour chaque feuille d'un arbre).

On suppose que l'on dispose d'une fonction **alea( )** sans paramètre qui renvoie un nombre réel (au sens informatique) aléatoire (loi de probabilité uniforme) entre 0 et 1 et d'une fonction **part-ent (p)** qui renvoie la partie entière du réel **p**.

Les algorithmes seront écrits dans le langage Pascal, ou dans un langage algorithmique proche de Pascal ou encore dans le langage Fortran. Dans ce dernier cas, les pointeurs seront des indices dans des tableaux.

1. **Énumération des arbres binaires.**

   1.1. Montrer que pour tout mot $f \in D$, le nombre de a dans f est égal au nombre de b dans f.

   1.2. Montrer que pour tout mot $f \in D$ et pour tout facteur gauche g de f, le nombre de a dans g est supérieur ou égal au nombre de b dans g.

   1.3. Montrer qu'il existe une bijection entre l'ensemble des arbres binaires à $2n + 1$ sommets ($n \geqslant 0$) et l'ensemble des mots de D à $2n$ lettres.

   1.4. Montrer que, pour tout $n \geqslant 0$ donné, il existe une bijection ϕ_n entre l'ensemble des couples (f, m) où f est un mot de longueur $2n$ de D et m un entier tel que $n \geqslant m \geqslant 0$ et l'ensemble des mots sur A qui ont n lettres a et n lettres b.

   Idée : m peut désigner une lettre a dans f. On peut alors faire la réduction des facteurs ab en 1 jusqu'à ce que cela ne soit plus possible, à condition que la lettre a du facteur ab ne soit pas la lettre a désignée par m. Dans le mot restant, on change les lettres a en b et inversement.

   1.5. En déduire que le nombre d'arbres binaires à $2n + 1$ sommets est égal à $\dfrac{(2n)!}{n!\,(n+1)!}$.

2. **Génération d'un arbre binaire aléatoire.**

   2.1. Donner l'algorithme de la procédure **genere(tab,i,p,q)** qui génère un mot contenant **p** lettres a et **q** lettres b dans le tableau **tab** à partir de l'indice **i** selon la loi uniforme (chaque mot de longueur $p + q$ a la même probabilité).

   Rappel : la probabilité pour qu'un tel mot commence par a (et soit donc suivi d'un mot contenant $p - 1$ lettres a et q lettres b) est $\dfrac{p}{p+q}$. La probabilité pour qu'un tel mot commence par b (et soit donc suivi d'un mot contenant p lettres a et $q - 1$ lettres b) est $\dfrac{q}{p+q}$.

   2.2. Donner l'algorithme de la procédure **traduction(tab,n, ,res)** qui, à partir d'un mot de longueur **2n** (dans le tableau **tab**) ayant **n** lettres a et **n** lettres b donne, dans **res**, le mot de Dyck correspondant de longueur **2n**.

   2.3. Donner l'algorithme de la procédure **construct(tab,n,point)** qui construit l'arbre binaire pointé par le pointeur **point** correspondant au mot de Dyck de longueur **2n** du tableau **tab**. On pourra utiliser une fonction **alloue()** sans paramètre qui alloue une cellule de type sommet et renvoie un pointeur sur cette cellule.

   2.4. Donner l'algorithme de la procédure **tirage (n,point)** qui donne un arbre binaire aléatoire ayant **2n + 1** sommets et pointé par le pointeur **point.**

   2.5. Montrer que cet algorithme produit un arbre binaire aléatoire.