MATHEMATIQUES DE L'INFORMATIQUE

Le problème porte sur l'analyse d'algorithmes pour l'évaluation d'expressions arithmétiques et l'allocation de registres en compilation. La partie I introduit les résultats de base de combinatoire des arbres. La partie II décrit une méthode d'évaluation (dite ***méthode d'allocation gauche-droite***) et les résultats de dénombrement correspondants. La partie III présente l'analyse asymptotique des résultats de la partie II. La partie IV a pour objet une autre méthode d'évaluation (dite ***méthode d'Ershov***).

RAPPELS ET DEFINITIONS GENERALES

Soient $\Omega = \{\omega_1,\omega_2,\cdots,\omega_k\}$ et $X = \{x_1,x_2,\cdots,x_l\}$ deux ensembles finis disjoints. L'ensemble Ω est appelé ensemble des *opérateurs* (dans ce problème, il s'agira d'opérateurs binaires) et l'ensemble X est appelé ensemble des *variables*. On considère l'ensemble $\mathbf{B}(\Omega;X)$ des arbres binaires dont les sommets binaires _ ou sommets *internes* _ sont étiquetés par des éléments de Ω, et dont les sommets zero-aires _ ou sommets *externes* _ sont étiquetés par des éléments de X. Les arbres considérés sont représentables graphiquement (voir exemple 1). Ils peuvent être définis récursivement ou spécifiés par une définition dans le langage de programmation Pascal (les deux définitions sont équivalentes).

A. Définition récursive. L'ensemble $\mathbf{B}(\Omega;X)$ des arbres binaires est défini par les conditions suivantes:

- chaque variable $x \in X$ est un arbre;
- si t_1 et t_2 sont des arbres et si $\omega \in \Omega$ est un opérateur, alors le triplet $t = <\omega,t_1,t_2>$ est un arbre; les arbres t_1 et t_2 sont appelés respectivement sous-arbres gauche et droit de l'arbre t.

B. Définition dans le langage de programmation Pascal. On définit le type *sommet* comme un enregistrement avec variante. Il y a deux catégories de sommets: les sommets de catégorie *interne* comportent un champ *operateur* de type Ω, et deux champs *gauche* et *droite* qui sont des pointeurs sur d'autres sommets; les sommets de catégorie *externe* sont formés d'un unique champ *variable* de type X. Le type *arbre* est défini comme pointeur sur un sommet. Les types *sommet* et *arbre* sont ainsi spécifiés par:

```
type intext = (interne, externe);
     arbre = ↑sommet;
     sommet = record
                    case categorie : intext of
                         interne : (operateur : Ω; gauche, droite : arbre);
                         externe : (variable : X);
                    end;
```

Une structure chainée a de type ***arbre*** appartient à $\mathbf{B}(\Omega;X)$ si et seulement si: (i) le graphe non orienté associé est sans cycle, et (ii) du sommet sur lequel pointe a on peut atteindre tout autre sommet par une succession de pointeurs gauches ou droits.

Le sommet $a\uparrow$ sur lequel pointe a est la *racine* de l'arbre a; les arbres $a\uparrow.gauche$ et $a\uparrow.droite$ sont les sous-arbres gauche et droit respectivement de l'arbre a.

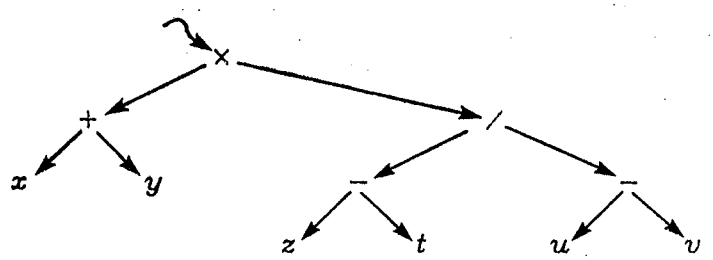
On appelle ***taille*** d'un arbre le nombre de sommets internes qu'il comprend et l'on dénote par $\mathbf{B}_n(\Omega,X)$ l'ensemble des arbres de $\mathbf{B}(\Omega;X)$ de taille n.

Un arbre de $\mathbf{B}(\Omega;X)$ est représentable graphiquement dans le plan. On notera qu'alors les arbres gauche et droit issus d'un sommet sont distingués par la représentation planaire (voir l'exemple 1).

A tout arbre de $\mathbf{B}(\Omega;X)$ est associé par lecture préfixe une expression bien parenthésée sur l'ensemble des opérateurs binaires Ω et sur l'ensemble des variables X. On supposera par la suite pour simplifier la description des algorithmes que les éléments de Ω et X sont des caractères au sens du langage de programmation Pascal:

```
type Ω =char;
     X =char;
```

Exemple 1: La structure chainée suivante



est un arbre de $\mathbf{B}_5(\Omega,X)$ où $\Omega=\{+,-,\times,/\}$ et $X=\{x,y,z,t,u,v\}$ auquel correspond l'expression bien parenthésée:

$$((x+y)\times((z-t)/(u-v))) .$$

NOTATIONS:

Dans la suite du problème, on note $\log x$ le logarithme népérien de x. On étend la définition des coefficients binomiaux en posant:

- $\binom{m}{n} = \frac{m!}{n!(m-n)!}$ lorsque m et n sont des entiers qui vérifient $0\leq n\leq m$;
- $\binom{m}{n} = 0$ dans les autres cas.

Partie I

Eléments de combinatoire des arbres.

1. On désigne par $b_n^{<k,l>}$ le nombre d'éléments de $\mathbf{B}_n(\Omega,X)$ (k et l sont les cardinaux respectifs de Ω et X). Exprimer en fonction de n, k et l le rapport:

$$\frac{b_n^{<k,l>}}{b_n^{<1,1>}} .$$

2. On pose $b_n = b_n^{<1,1>}$. Montrer que les b_n vérifient la récurrence:

$$b_n = \sum_{j=0}^{n-1} b_j b_{n-1-j} \qquad (n\geq 1)$$

et que $b_0 = 1$.

3. Soient $\Omega_1 = \{\omega\}$ et $X_1 = \{x\}$. On définit une application π de $\mathbf{B}(\Omega_1;X_1)$ dans l'ensemble $\{\omega,x\}^*$ des mots sur l'alphabet $\{\omega,x\}$ par les règles suivantes:

- $\pi(t) = x$ si t est de taille 0;
- $\pi(t) = \omega.\pi(t_1).\pi(t_2)$ si t est de taille ≥ 1 et si t_1 et t_2 sont les sous-arbres gauche et droit de t respectivement (la notation $u.v$ représente ici la concaténation des mots u et v).

On définit la fonction δ de $\{\omega,x\}^*$ dans l'ensemble $\mathbf{Z}$ des entiers relatifs par:

$$\delta(w) = |w|_\omega - |w|_x$$

où $|w|_y$ représente le nombre d'occurrences de la lettre y dans le mot w.

On désigne par L l'image par π de $\mathbf{B}(\Omega_1;X_1)$. Montrer que tout mot de L vérifie les deux propriétés:

(P1) $\delta(w) = -1$;

(P2) pour tout préfixe strict u de w: $\delta(u)\geq 0$.

On rappelle que u est préfixe strict de w s'il existe un mot v non vide tel que $w = u.v$.

Montrer que l'application π est une injection de $\mathbf{B}(\Omega_1;X_1)$ dans $\{\omega,x\}^*$. En déduire un minorant strictement positif R_1 du rayon de convergence de la

série:

$$\sum_{n=0}^{\infty} b_n z^n .$$

4. Soit $b(z)$ la somme de la série $\sum_{n=0}^{\infty} b_n z^n$ lorsque $|z|<R_1$; montrer que $b(z)$ vérifie l'équation:

$$b(z) = 1 + zb^2(z) .$$

5. Montrer que

$$b_n = \frac{1}{n+1}\binom{2n}{n} .$$

Déterminer des nombres réels A, λ, ρ tels que:

$$\lim_{n\to\infty} \frac{b_n}{\lambda A^n n^\rho} = 1 .$$

On pourra pour cela utiliser sans la démontrer la formule de Stirling:

$$\lim_{n\to\infty} \frac{n!}{n^n e^{-n}\sqrt{2\pi n}} = 1 .$$

Déterminer le rayon de convergence de la série $\sum_{n=0}^{\infty} b_n z^n$.

Partie II

Allocation Gauche-Droite et résultats de dénombrements.

Un *programme élémentaire* est une suite d'*instructions* sur les variables de l'ensemble X et sur les éléments d'un tableau de dimension infinie dit *tableau des registres*

$$R = R[0], R[1], R[2], R[3], \cdots$$

Ces instructions utilisent les opérateurs de Ω. Elles sont de l'un des deux types suivants:

$$R[i] := x ; \qquad x\in X,\ i\geq 0 \qquad (I)$$

$$R[i] := R[i']\,\omega\,R[i'']; \qquad \omega\in\Omega,\ i,i',i''\geq 0 \qquad (II)$$

On notera que dans les définitions (I), (II), il n'est pas imposé aux indices i, i', i'' d'être distincts.

On dira qu'un programme élémentaire Π *utilise* r registres si et seulement si r est le plus grand indice de registre qui intervient dans le programme Π.

On appelle résultat de Π le contenu du registre $R[0]$ à l'issue de l'exécution du programme Π. Ce contenu est défini conformément à la sémantique usuelle de l'affectation. On suppose les variables de X initialisées, mais non les éléments du tableau R. Ces notions sont illustrées par l'exemple 2 ci-dessous.

On définit alors la procédure *genere*, dans laquelle *decalage* est une variable externe à la procédure et de type *integer*:

```
procedure genere (a : arbre);
begin
if a↑.categorie =externe
    then writeln ('R[',decalage,']:=',a↑.variable,';')
    else begin
    genere (a↑.gauche); decalage :=decalage +1;
    genere (a↑.droite); decalage :=decalage −1;
    writeln ('R[',decalage,']:=R[',decalage,']',a↑.operateur,'R[',decalage +1,'];');
    end
end;
```

On définit enfin une application χ de l'ensemble $\mathbf{B}(\Omega;X)$ dans l'ensemble des entiers naturels par les règles suivantes:

- si t est de taille 0, alors $\chi(t)=0$;
- si t est de taille ≥ 1 et si t_1 et t_2 sont les sous-arbres gauche et droit de t respectivement, alors:

$$\chi(t) = \max(\chi(t_1), 1+\chi(t_2)) .$$

Exemple 2: Le programme élémentaire suivant

```
R[0]:=z;
R[1]:=t;
R[0]:=R[0]−R[1];
R[1]:=u;
R[2]:=v;
R[1]:=R[1]−R[2];
R[0]:=R[0]/ R[1];
```

avec Ω et X définis comme dans l'exemple 1 a pour résultat:

$$((z-t)/(u-v)) ;$$

il utilise 2 registres. Pour l'arbre a associé à cette expression, $\chi(a)=2$.

6. Donner sans démonstration le résultat du programme *genere* lorsqu'il est appliqué à l'arbre de l'exemple 1, la variable *decalage* étant initialisée à la valeur 0.

Montrer que l'exécution de la séquence

decalage :=0; *genere* (*a*);

produit un programme élémentaire dont le résultat est l'expression bien parenthésée associée à a, et qui utilise $\chi(a)$ registres.

7. On note $c_{n,h}^{<k,l>}$ le nombre d'éléments de $\mathbf{B}_n(\Omega,X)$ dont le paramètre χ est inférieur ou égal à h, et l'on pose $c_{n,h} = c_{n,h}^{<1,1>}$.

Montrer que le rapport

$$\frac{c_{n,h}^{<k,l>}}{b_n^{<k,l>}}$$

est indépendant de k et l.

Déterminer les quantités $c_{4,h}$ pour $h=0,1,2,3,4,5,6$ (la rédaction des réponses n'est pas exigée mais on demande de dessiner les arbres correspondants à chacun des 7 cas).

Déterminer les valeurs extrémales de χ sur $\mathbf{B}_n(\Omega,X)$. On pourra commencer par considérer le cas des arbres dits *filiformes* dans lesquels tout sommet interne possède au plus un successeur gauche ou un successeur droit qui soit sommet interne.

8. Montrer que les fonctions

$$c_h(z) = \sum_{n\geq 0} c_{n,h} z^n$$

sont analytiques dans le disque de convergence de $b(z)$, et vérifient la recurrence:

$$c_h(z) = \frac{1}{1-z\,c_{h-1}(z)} \qquad (h\geq 1)$$

avec $c_0(z)=1$. (On pourra pour cela reprendre un raisonnement semblable à celui utilisé pour traiter les questions 2 et 4).

9. Montrer qu'il existe une unique suite de polynomes $F_1(z)$, $F_2(z)$, $\cdots$ vérifiant les conditions:

$$F_1(z) = 1\ ; \tag{C1}$$

$$c_h(z) = \frac{F_{h+1}(z)}{F_{h+2}(z)} \qquad (h\geq 0)\ ; \tag{C2}$$

$$c_h(z)-c_{h-1}(z) = \frac{z^h}{F_{h+1}(z)F_{h+2}(z)} \qquad (h\geq 1)\ . \tag{C3}$$

Etablir que la suite $\{F_h(z)\}_{h\geq 1}$ vérifie une récurrence linéaire à coefficients polynomiaux en z. Montrer que le coefficient de z^n dans $F_h(z)$ est de la forme:

$$(-1)^\alpha \binom{\beta}{\gamma}$$

pour des entiers α, β, γ dépendant de h et n dont on donnera l'expression explicite.

10. Montrer que pour ϑ n'appartenant pas à $\{\frac{\pi}{2}\}\mathbf{Z}$ et $h\geq 1$:

$$F_h\left(\frac{1}{4\cos^2\vartheta}\right) = \frac{1}{(2\cos\vartheta)^{h-1}}\,\frac{\sin(h\vartheta)}{\sin\vartheta}\ .$$

Déterminer les racines de F_h et établir pour $n\geq 1$ et $h\geq 1$ l'expression

$$c_{n,h} = \frac{4^{n+1}}{h+2} \sum_{1\leq j\leq \frac{(h+1)}{2}} \sin^2\left(\frac{j\pi}{h+2}\right)\cos^{2n}\left(\frac{j\pi}{h+2}\right)\ .$$

11. On pose $\varphi(u) = \dfrac{u}{(1+u)^2}$ où u est une variable complexe. On désigne par Γ l'image par φ du cercle $|u| = \dfrac{1}{100}$ parcouru dans le sens direct. Calculer l'intégrale:

$$\frac{1}{2i\pi}\int_\Gamma \frac{dz}{z}\ .$$

On pose $z=\varphi(u)$. Exprimer en fonction de u la quantité

$$b(z)-c_h(z)$$

lorsque $|u|\leq\frac{1}{100}$.

12. Montrer que

$$b_n-c_{n,h}=\sum_{j\geq 1}\binom{2n}{n+1-j(h+2)}-2\binom{2n}{n-j(h+2)}+\binom{2n}{n-1-j(h+2)}.$$

On pourra pour cela observer que

$$b_n-c_{n,h}=\frac{1}{2i\pi}\int_\Gamma[b(z)-c_h(z)]\frac{dz}{z^{n+1}}.$$

Partie III

Analyse asymptotique.

On définit les quantités $p_{n,h}$, $q_{n,h}$ et $\bar{\chi}_n$ par:

$$p_{n,h}=\frac{c_{n,h}-c_{n,h-1}}{b_n}\;;\quad q_{n,h}=\frac{c_{n,h}}{b_n}$$

$$\bar{\chi}_n=\sum_{h\geq 1}h\,p_{n,h}\,.$$

On observera sans démonstration, par suite des résultats de la question 7, que $p_{n,h}$ représente la probabilité qu'un arbre formé de n sommets internes ait un paramètre χ de valeur h, sous l'hypothèse que les $b_n^{<k,l>}$ arbres de $\mathbf{B}_n(\Omega,X)$ ont même probabilité (égale à $\frac{1}{b_n^{<k,l>}}$); sous cette même hypothèses, $\bar{\chi}_n$ représente l'espérance du paramètre χ.

13. Montrer que:

$$\bar{\chi}_n=\sum_{h\geq 0}[1-q_{n,h}].$$

14. Montrer que, pour tout x vérifiant $0\leq x\leq\frac{\pi}{2}$, on a:

$$\cos x\leq\exp(-\frac{x^2}{2}).$$

Etablir pour $n\geq 1$ et $h\geq 0$ la majoration:

$$c_{n,h}\leq\frac{1}{2}4^{n+1}(\cos\frac{\pi}{h+2})^{2n}.$$

Montrer qu'il existe deux constantes réelles $K_1>0$ et $\gamma_1>0$ telles que, pour tout $n\geq 2$ et tout h vérifiant $2\leq h+2\leq\sqrt{n}/\log n$, on ait:

$$q_{n,h}\leq K_1\exp(-\gamma_1\log^2 n).$$

15. Montrer que pour tout $x\geq 0$, on a:

$$\frac{x}{1+x}\leq\log(1+x)\leq x.$$

On pose pour $0 \leq j \leq n$:

$$Q(n,j) = \frac{\binom{2n}{n}}{\binom{2n}{n-j}}$$

Etablir pour tout $n \geq 1$ et $0 \leq j \leq n$ l'encadrement:

$$\frac{j^2}{n+j} \leq \log Q(n,j) \leq \frac{j^2}{n-j+1} .$$

16. Montrer que pour $n \geq 2$ et $h \geq 1$, on a:

$$1 - q_{n,h} \leq (n+1) \sum_{j=h+1}^{n} \frac{1}{Q(n,j)} .$$

Montrer qu'il existe deux constantes réelles $K_2 > 0$ et $\gamma_2 > 0$ telles que pour tout $n \geq 2$ et $h \geq \sqrt{n} \log n$, on ait:

$$1 - q_{n,h} \leq K_2 \exp(-\gamma_2 \frac{h^2}{n}) .$$

17. Montrer qu'il existe deux constantes réelles $L_1 > 0$ et $L_2 > 0$ telles que pour tout $n \geq 2$, $\overline{\chi}_n$ vérifie l'encadrement:

$$L_1 \frac{\sqrt{n}}{\log n} < \overline{\chi}_n < L_2 \sqrt{n} \log n .$$

18. On définit la fonction $\Theta(x)$ pour x réel, $x \neq 0$, par:

$$\Theta(x) = \sum_{k \geq 1} e^{-k^2 x^2} (4k^2 x^2 - 2) ,$$

avec par convention $\Theta(0) = 1$. On admettra sans démonstration que $\Theta(x)$ est indéfiniment différentiable sur $[-\infty; +\infty]$.
Montrer que la série:

$$\sum_{h \geq 1} \Theta(\frac{h}{\sqrt{n}})$$

converge pour tout $n \geq 1$; soit σ_n sa somme.
Montrer que l'intégrale

$$\int_0^{\infty} \Theta(\lambda) \, d\lambda .$$

est absolument convergente; soit K sa valeur.
Démontrer que

$$\lim_{n \to \infty} \frac{\sigma_n}{\sqrt{n}} = K .$$

19. On définit l'intégrale:

$$\Psi(s) = \int_0^{\infty} \Theta(x) x^{s-1} \, dx .$$

Montrer que $\Psi(s)$ est définie et continue pour s réel, $s > 0$. Montrer que pour $s > 1$, on a:

$$\Psi(s) = (s-1) \zeta(s) \Gamma(\frac{s}{2}) ,$$

où les fonctions $\zeta(s)$ et $\Gamma(s)$ sont définies pour s réel par:

$$\zeta(s) = \sum_{k\geq 1} \frac{1}{k^s} \qquad (s>1)\,,$$

$$\Gamma(s) = \int_0^{\infty} e^{-x} x^{s-1}\, dx \qquad (s>0)\,.$$

Evaluer $\Psi(1)$ et en déduire la valeur de K.
On pourra pour cela admettre les propriétés suivantes:

$$\lim_{s\to 1^+}(s-1)\zeta(s) = 1$$

$$\Gamma(\frac{1}{2}) = \sqrt{\pi}\,.$$

20. Montrer que si $\{h_n\}_{n\geq 1}$ est une suite d'entiers positifs tels que

$$\lim_{n\to\infty} \frac{h_n}{\sqrt{n}} = \lambda$$

pour un certain réel λ, $\lambda>0$, alors:

$$\lim_{n\to\infty}(1-q_{n,h_n}) = \mathcal{P}(\lambda)$$

Pour la suite du problème, on admettra sans démonstration que lorsque $n\to\infty$:

$$\lim_{n\to\infty} \frac{\bar{X}_n}{\sigma_n} = 1$$

Partie IV

Méthode d'Ershov.

On définit une application ρ de $\mathbf{B}(\Omega;X)$ dans l'ensemble des entiers naturels par les règles suivantes:

- si t est de taille 0, alors $\rho(t)=0$;
- si t est de taille ≥ 1 et si t_1 et t_2 sont les sous arbres gauche et droit respectivement de t, alors

$$\rho(t) = \begin{cases} \max(\rho(t_1),\rho(t_2)) & \text{si } \rho(t_1)\neq\rho(t_2) \\ 1+\rho(t_1) & \text{si } \rho(t_1)=\rho(t_2) \end{cases}$$

On suppose que la fonction (au sens du langage de programmation Pascal) *rho* de type:

function *rho*(*a*:*arbre*):*integer*;

retourne la valeur entière $\rho(a)$ lorsqu'elle est appliquée à un argument a de type *arbre*. On définit alors la procédure *genere*2 par:

```
procedure genere2(a:arbre);
begin
if a↑.categorie =externe
    then writeln('R[',decalage,']:=',a↑.variable)
    else
    if rho(a↑.gauche)>rho(a↑.droite)
        then begin
        genere 2(a↑.gauche); decalage :=decalage +1;
        genere 2(a↑.droite); decalage :=decalage −1;
        writeln ('R[',decalage,']:=R[',decalage,']',a↑.operateur,'R[',decalage +1,'];');
        end
        else begin
        genere 2(a↑.droite); decalage :=decalage +1;
        genere 2(a↑.gauche); decalage :=decalage −1;
        writeln ('R[',decalage,']:=R[',decalage +1,']',a↑.operateur,'R[',decalage,'];');
        end
end;
```

21. Donner sans démonstration le résultat du programme *genere2* lorsqu'il est appliqué à l'arbre de l'exemple 1, la variable ***decalage*** étant initialisée à la valeur 0.

Montrer que l'execution de la séquence:

decalage :=0; *genere* 2(a);

produit un programme élémentaire dont le résultat est l'expression bien parenthésée associée à a et qui utilise $\rho(a)$ registres.

22. On désigne par $f_{n,v}^{<k,l>}$ le nombre d'éléments de $\mathbf{B}_n(\Omega,X)$ dont le paramètre ρ vaut v. Montrer que le rapport:

$$\frac{f_{n,v}^{<k,l>}}{b_n^{<k,l>}}$$

est indépendant de k et l; soit $r_{n,v}$ ce rapport. Montrer qu'il existe deux suites d'entiers $\{\lambda_n\}_{n\geq 0}$ et $\{\mu_n\}_{n\geq 0}$ avec $\lambda_n \leq \mu_n$ telles que:

$$r_{n,v} = 0 \quad \text{si } v \notin [\lambda_n ; \mu_n]$$
$$r_{n,v} \neq 0 \quad \text{si } v \in [\lambda_n ; \mu_n]$$

et déterminer explicitement les λ_n et μ_n.

On pourra pour cela considérer d'abord le cas des arbres filiformes définis à la question 7 puis le cas des arbres parfaits qui sont tels que tous les chemins joignant la racine de l'arbre à un sommet externe comportent le même nombre de pointeurs.

23. On pose:

$$\bar{\rho}_n = \sum_{v\geq 0} v\, r_{n,v}$$

Montrer l'existence de

$$\lim_{n\to\infty} \frac{\bar{\rho}_n}{\bar{\chi}_n}$$

et évaluer cette limite.

24. Discuter l'efficacité relative de la *méthode d'allocation gauche-droite* et de la *méthode d'Ershov* vis à vis des critères suivants:

- ordre de grandeur des temps de calcul des procédures *genere* et *genere2*,
- nombre de registres utilisés par les programmes élémentaires générés par ces procédures à partir d'un arbre de taille n.

On tiendra compte dans cette discussion du temps de calcul de la procédure *rho*.

Peut-on trouver une procédure qui, étant donné un arbre a de taille n, calcule un programme élémentaire Π dont le résultat est l'expression bien parenthésée associée à a et telle que:

- Π utilise $\rho(a)$ registres;
- le temps de calcul de la procédure est $O(n)$?

RAPPORT SUR L'EPREUVE
MATHEMATIQUES DE L'INFORMATIQUE

1. Description du problème

Le problème portait sur l'étude et l'évaluation comparée de 2 méthodes d'allocation de registre en compilation. La partie I présentait les éléments de combinatoire des arbres préparant à la suite du problème (relation à la notation "polonaise" préfixe, dénombrements de base). La partie II contenait la première stratégie d'évaluation dite gauche-droite, suivie de résultats de dénombrement (avec une brève incursion dans l'analyse complexe aux questions Q.11-12). La partie III avait pour objet l'étude de la distribution et de la moyenne du coût mesuré en nombre de registres de la méthode gauche-droite (au moyen de divers encadrements d'analyse réelle). La partie IV présentait une méthode alternative découverte par Ershov vers 1958, et l'on se contentait d'extraire les bornes extrêmales de son comportement vis-à-vis du nombre de registres générés, ce qui suffisait pour établir la comparaison.

Il convient d'abord de souligner que le problème traitait d'algorithmes et de programmes (écrits en Pascal) qui génèrent eux-mêmes des programmes : c'est là le rôle d'un compilateur qui doit notamment dans sa partie dite "génération de code" transformer des expressions (ici des expressions arithmétiques déjà données sous forme arborescente) en un programme élémentaire destiné à être exécuté ultérieurement. Ainsi le résultat des programmes *genere* et *genere2* est-il un *programme* qui évalue et non pas une évaluation directe. Quelques-uns des candidats qui ont abordé les questions parallèles 6 ou 21 n'ont pas perçu ce point (et l'un d'eux a même cru à une erreur d'énoncé!).

Ceci étant rappelé, l'efficacité de procédures de génération comme *genere* ou *genere2* se mesure naturellement par le nombre de registres utilisés (dans la pratique informatique le nombre de registres est limité et une mauvaise évaluation conduit lors de l'exécution à des rangements/chargements de résultats intermédiaires qui peuvent être coûteux).

Dans ce cadre, on observe - et c'était le but de la dernière question (Q.24) d'amener les candidats à cette conclusion - que :

> La procédure gauche-droite (*genere*) est simple à mettre en oeuvre (Q.6), ne nécessite qu'une passe sur l'arbre, et donc possède un coût en temps linéaire $O(n)$ en la taille n de l'arbre donné. Le nombre de registres utilisés est en moyenne $\sim \sqrt{\pi n}$ (Q.8-20).

La procédure d'Ershov (*genere2*) utilise le calcul de la fonction ρ, d'où (voir *infra* Q.24) un coût $O(n^2)$. Le nombre de registres utilisés est compris dans tous les cas entre 1 et $\sim\log_2 n$ (Q.22.23).

Comme l'a justement résumé un candidat : la méthode gauche-droite privilégie le temps de compilation alors que la méthode d'Ershov, dans sa forme *genere2* , privilégie l'exécution ultérieure du programme généré. (A noter cependant qu'une modification simple de la procédure *genere2*, opérant en 2 passes permet de concilier l'avantage du nombre de registres de la procédure *genere2* avec un temps de calcul linéaire $O(n)$).

2. Commentaire sur les copies

Le problème comprenait environ 1/3 des questions (Q.1-3,6-7,21-24) traitant de la combinatoire et de l'algorithmique des arbres, le reste se répartissant entre questions "algébriques" (surtout Q.4-5,8-10) et "analytiques" (Q.11-20) pour les évaluations correspondantes.

Hormis les copies blanches, la plupart des candidats ont également abordé les questions non-analytiques. Il est encourageant de constater pour cette première année d'existence de l'option que les candidats n'ont pas fuit les questions de nature combinatoire et algorithmique. Un certain nombre (environ 1/3 des copies notées 5) paraît avoir saisi l'esprit du problème. La partie III, plus analytique, n'a été qu'effleurée dans la plupart des copies.

Les divers groupes de questions étaient assez indépendants, et il nous semble que tout candidat connaissant les bases du programme de mathématiques du concours et des bases élémentaires de programmation (notion d'arbre et de procédure récursive) devait, s'il utilisait l'intégralité des 6 heures de l'épreuve, pouvoir obtenir une note au moins égale à 15.

Les correcteurs ont, pour la notation, tenu grand compte du soin apporté à la rédaction des copies.

La répartition des notes, sur 79 copies corrigées, est la suivante :

0	: 15 copies		20 à 24	: 7 copies	
1 à 4	: 23	"	25 à 29	: 4	"
5 à 9	: 10	"	30 à 34	: 4	"
10 à 14	: 5	"	35 à 39	: 2	"
15 à 19	: 8	"	40	: 1	"

3. Commentaires sur les questions

Partie I

Le nombre d'arbres de taille n vaut (pour k=1=1) le "nombre de Catalan" : $b_n = \frac{1}{n+1}\binom{2n}{n}$. *On établit ce*

résultat classique par la méthode des séries génératrices. Les premières questions ont pour objet de montrer qu'on peut se dégager de l'étiquetage et que la série génératrice existe en tant que fonction analytique.

Q1. Le rapport vaut $k^n 1^{n+1}$ car il représente le nombre de façons d'étiqueter un arbre formé de n sommets internes (donc de $n+1$ sommets externes, l'arbre étant binaire).

Q2. Un arbre de taille $n \geq 1$ comporte une racine, un sous arbre gauche de taille j avec $0 \leq j < n$ et un sous-arbre droit de taille $n-1-j$. Il suffit de compter les possibilités.

Q3. Les propriétés (P1) et (P2) se montrent par récurrence sur la taille des arbres. Pour l'injectivité, on peut remarquer que si $\pi(t)=\pi(t')$, alors t et t' ont même taille, puis raisonner par récurrence sur la taille des arbres en utilisant (P1) et (P2). On en déduit que b_n est majoré par le nombre de mots de longueur $2n+1$ terminés par un x, d'où $b_n \leq 2^{2n} 4^n$ et $R_1 = 1/4$.

Q4. Utiliser l'égalité établie à la question Q2.

Q5. Résoudre l'équation du second degré $b(z)=1+zb^2(z)$, d'où :

$$b(z) = \frac{1-\sqrt{1-4z}}{2z}$$

L'autre racine doit être écartée car non analytique en 0. (Quelques candidats ont clairement justifié ce choix). On trouve l'expression de b_n par développement de $(1-4z)^{1/2}$, puis par la formule de Stirling : $b_n \sim \frac{1}{\sqrt{\pi}} 4^n n^{-3/2}$. Le rayon de convergence de $b(z)$ est donc exactement 1/4.

Partie II

La procédure genere alloue les registres lors d'une simple lecture de gauche à droite de l'arbre d'expression. Le nombre de registres du programme généré se caractérise par un paramètre χ des arbres binaires. Il en résulte une expression explicite des séries génératrices associées, qui sont des fractions rationnelles. De là, on obtient d'une part une forme "trigonométrique", d'autre part une forme "binomiale" des dénombrements liés au paramètre χ.

Q6. Appliqué à l'arbre de l'exemple 1, le programme donne :

```
R[0]:=x ;
R[1]:=y ;
R[0]:=R[0]+R[1] ;
R[1]:=z ;
R[2]:=t ;
R[1]:=R[1]-R[2] ;
R[2]:=u ;
R[3]:=v ;
R[2]:=R[2]-R[3] ;
R[1]:=R[1]/R[2] ;
R[0]:=R[0]*R[1] ;
```

On prouve simplement par récurrence sur la taille de l'arbre , l'assertion suivante :

Pour tout $i \geq 0$:

(a) L'exécution de la séquence :

decalage:=i;genere(a) ; (S)

produit un programme tel qu'après exécution, le contenu du registre $R[i]$ est l'expression bien parenthésée associée à l'arbre *a*.

(b) A l'issue de l'exécution de (*S*), *decalage* vaut i et lors de l'exécution *decalage*$\geq i$.

(c) Dans le programme obtenu seuls interviennent les registres $R[i], R[i+1], \ldots, R[i+\chi(a)]$.

Peu de candidats ont trouvé la forme correcte de l'hypothèse de récurrence, plus générale que l'énoncé, qui nécessitait une bonne compréhension de la procédure *genere*.

Q7. Il y a 14 arbres de taille 4 (*cf.*Q5) et l'on trouve :

$$c_{4,0}=0,\ c_{4,1}=1,\ c_{4,2}=8,\ c_{4,3}=13,\ c_{4,h}\ b_4=14$$

pour $h \geq 4$. (Il y a eu beaucoup d'erreur sur cette question facile alors qu'on pouvait s'assurer de la cohérence avec la forme : $b_4 = \frac{1}{5}\binom{8}{4} \equiv 14$).

Q8. La formule cherchée résulte de la relation de récurrence :

$$c_{n,h} = \sum_{j=0}^{n} c_{h,j} \cdot c_{h-1,n-1-j}$$

(un arbre de paramètre χ au plus égal à h est formé d'un sous-arbre gauche de paramètre au plus égal à h et d'un sous-arbre droit de paramètre au plus égal à h-1). En passant à la fonction génératrice comme précédemment, on trouve :

$$c_h(z) = 1 + z c_h(z) c_{h-1}(z).$$

Q9. La récurrence sur les $c_h(z)$ de Q8. montre que si :

$$c_{h-1}(z) = \frac{F_h(z)}{F_{h+1}(z)} \quad \text{alors} \quad c_h(z) = \frac{F_{h+1}(z)}{F_{h+1}(z) - zF_h(z)}$$

d'où la récurrence linéaire sur les F_h :

$$F_{h+2}(z) = F_{h+1}(z) - zF_h(z)$$

On vérifie alors facilement par récurrence l'identité :

$$F_{h+1}^2 - F_h F_{h+2} = z^h$$

d'où la forme de $c_h - c_{h-1}$. (Les candidats ont souvent mal dégagé les hypothèses de récurrence, et mal distingué les preuves d'existence et d'unicité).

En construisant la table des coefficients des premiers F_h, on s'aperçoit que le coefficient de z^n dans $F_h(z)$ vaut :

$$(-1)^n \binom{h-n-1}{n}$$

propriété qui se vérifie par une récurrence analogue à celle des coefficients binomiaux. Le degré de $F_h(z)$ est : $\left|\frac{h-1}{2}\right|$.

Q.10. La forme "trigonométrique" de F_h s'établit sans difficulté par récurrence sur h. On en déduit les $\left|\frac{h-1}{2}\right|$ racines distinctes de $F_h(z)$ qui sont :

$$\zeta_j = \frac{1}{4\cos^2 \frac{j\pi}{h}}$$

avec $0<j<h/2$.

La formule donnant les $c_{n,h}$ s'obtient en décomposant en éléments simples la fraction rationnelle $c_h(z)$:

$$c_h(z) = \sum_j \frac{\gamma_{h,j}}{z - \zeta_j}$$

(il fallait calculer les $\gamma_{h,j}$ en utilisant la forme "trigonométrique" des F_h), puis en développant en série entière les éléments simples obtenus. Peu de candidats ont trouvé ce principe.

Q11. Soit γ le cercle $|u| = 1/100$. On a :

$$\frac{1}{2i\pi} \int_\Gamma \frac{dz}{z} = \frac{1}{2i\pi} \int_\gamma \frac{\phi'(u)}{\phi(u)}\, du = \frac{1}{2i\pi} \int_\gamma \frac{1-u}{1+u}\, \frac{du}{u}$$

et la dernière intégrale vaut 1 par le théorème des résidus.

Pour u réel et $|u|\leqq 1/100$, $z=\phi(u)$ est réel avec $|z|<1/4$. On trouve qu'alors $b(z) = 1+u$. La résolution de la récurrence linéaire des F_h montre que :

$$F_h(z) = \frac{y_1^h - y_2^h}{y_1 - y_2}$$

avec $y_1 = \frac{1+\sqrt{1-4z}}{2}$ et $y_2 = \frac{1-\sqrt{1-4z}}{2}$. D'où par une substitution analogue la forme de $c_h(z)$ en fonction de u et finalement :

$$b(z) - c_h(z) = u^{h+1}\,\frac{1-u^2}{1-u^{h+2}}$$

(Le résultat s'étend pour $|u|\leqq 1/100$ par prolongement analytique).

Q12. Compte tenu de l'indication de l'énoncé et puisque Γ est d'indice 1, d'après Q11., on est ramené à évaluer l'intégrale :

$$b_n - c_{n,h} = \frac{1}{2i\pi}\int_{\gamma} \frac{(1+u)^{2n}(1-2u+u^2)u^{h-n}}{1-u^{h+2}}\,du$$

laquelle se calcule par résidus. Par exemple :

$$\frac{1}{2i\pi}\int_{\gamma} \frac{(1+u)^{2n}u^{h+2}\,du}{1-u^{h+2}\quad u^{n+1}} = \frac{1}{2i\pi}\int_{\gamma} \sum_{j\geqq 1} (1+u)^{2n}u^{j(h+2)}\,\frac{du}{u^{n+1}}$$

$$= \sum_{j\geqq 1} (n-j(h+2))$$

(Cette question relativement difficile n'a pas été traitée).

PARTIE III

L'utilisation de majorations élémentaires appliquées resp. aux formes trigonométriques et binomiales montre que : (a) de "petites" valeurs de χ, inférieures à $\sqrt{n}/\log n$, (b) de "grandes" valeurs de χ, supérieures à $\sqrt{n}\,\log n$, ont des probabilités exponentiellement petites. Des évaluations plus fines suggèrent l'existence d'une loi limite de distribution du paramètre χ exprimée par une fonction "theta". On trouve alors que la moyenne de χ sur l'ensemble des arbres de taille n vaut asymptotiquement $\sqrt{\pi n}$.

Q13. Il s'agit là de la forme classique de l'espérance d'une variable aléatoire discrète :

$$\overline{\chi}_n = (p_{n,1}+p_{n,2}+p_{n,3}+\ldots)+(p_{n,2}+p_{n,3}+\ldots)+(p_{n,3}+p_{n,4}+\ldots)+\ldots$$

(C'est un exemple de transformation d'Abel ou sommation par parties).

Q14. Considérer $f(x) = \cos x . e^{x^2/2}$ et observer que $f'(x) = \cos x(x-\mathrm{tg}x)e^{x^2/2}$ est négative sur l'intervalle.(Une preuve par développement de Taylor, plus longue, était possible). La majoration de la forme trigonométrique des $c_{n,h}$ est triviale (cosinus décroissant et sinus majoré par 1).

En lui appliquant la borne $\cos x \leqq e^{-x^2/2}$, on trouve :

$$q_{n,h} \leqq 2.\frac{4^n}{b_n} \exp\left(-\frac{n\pi^2}{(h+2)^2}\right),$$

et l'on conclut facilement avec $\gamma_1 = 9$ en utilisant la forme asymptotique de b_n.

Q15. Partir de :

$$\log Q(n,j) = \sum_{i=1}^{j} \log\left(1+\frac{j}{n+i-j}\right)$$

et appliquer à $\log(1+x)$ les bornes élémentaires.

Q16. On utilise cette fois-ci la forme binomiale de $b_n - c_{n,h}$ et l'on observe que :

$$\frac{b_n - c_{b,h}}{b_n} \leqq \sum_{h=h+1}^{n} \left[\frac{1}{Q(n,j(h+2)-1)} \quad \frac{1}{Q(n,j(h+2)+1)}\right]$$

On note alors qu'avec la question Q15 :

$$1-q_{n,h} \leqq (n+1) \sum_{h=h+1}^{n} e^{-j^2/(n+j)} \leqq (n+1) \sum_{j=h+1}^{\infty} d^{-j^2/(2n)}$$

$$\leqq (n+1) \sum_{J=h^2}^{\infty} e^{-J/(2n)} \leqq (n+1) \frac{e^{-h^2/(2n)}}{1-e^{-1/(2n)}}$$

On conclut en prenant $\gamma_2 = 1/2$.

Q17. On décompose la forme (Q13) de $\overline{\chi}_n$ en :

$$\overline{\chi}_n = \sum_{h<H} \left[1-q_{n,h}\right] + \sum_{h=H}^{n} \left[1-q_{n,h}\right]$$

Borne inférieure : On prend $H=H_1 \equiv \sqrt{n}/\log n - 2$; on minore les termes de la deuxième somme par 0, ceux de la première somme en utilisant (Q14) :

$$\overline{\chi}_n \geqq H_1 - K_1 n \exp(-\gamma_1 \log^2 n)$$

Borne supérieure : On prend $H=H_2 \equiv \sqrt{n}\log n$; on majore les termes de la première somme par 1, ceux de la deuxième somme par (Q16) :

$$\overline{\chi}_n \leqq H_2 + K_2 n \exp(-\gamma_2 \log^2 n)$$

Q18. La fonction Θ est à décroissance exponentielle à l'infini d'où résulte la convergence de la somme et de l'intégrale. On observe alors que $\sigma_n/\sqrt{n}$ est la somme de Riemann de pas uniforme $1/\sqrt{n}$ associée à l'intégrale. La somme de Riemann tend donc vers la valeur de l'intégrale. (Décomposer l'intégrale en 2 parties).

Q19. La fonction $\Psi(s)$ est définie car Θ est continue en 0 et à décroissance exponentielle à l'infini. La continuité se montre par le théorème de Legesgue. On observe que :

$$\int_0^\infty e^{-x^2}(4x^2-2)x^{s-1}dx = (s-1)\Gamma(\frac{s}{2})$$

$$\int_0^\infty e^{-k^2x^2}(4k^2x^2-2)x^{s-1}dx = k^{-s}(s-1)\Gamma(\frac{s}{2})$$

d'où par sommation sur k:

$$\Psi(s) = (s-1)\zeta(s)\Gamma(s)$$

Q20. [Indication] Soit :

$$D_{n,t} = \frac{1}{b_n}\left[\binom{2n}{n-t+1} - 2\binom{2n}{n-t} + \binom{2n}{n-t-1}\right]$$

On trouve (cf. Q.15) que si $t\to\infty$ avec $t/\sqrt{n}\to\mu$

$$D_{n,t} \to e^{-\mu 2}(4\mu^2-2)$$

Donc chaque terme de la forme binomiale de $1-q_{n,h}$ tend vers le terme correspondant de la somme définissant Θ. (Cette question qui montre l'existence d'une loi limite pour la distribution du paramètre χ n'a pas été traitée par les candidats).

PARTIE IV

La méthode d'Ershov consiste à toujours évaluer en premier lieu la sous-expression la plus "complexe" (en un sens donné par le paramètre "ρ"). Elle conduit à une meilleure utilisation des registres.

Q21. Le résultat de *genere 2* appliqué à l'exemple est :

```
R[0]:=v;
R[1]:=u;
R[0]:=R[1]-R[0];
R[1]:=t;
R[2]:=z;
R[1]:=R[2]-R[1];
R[0]:=R[1]-R[0];
R[1]:=y;
R[2]:=x;
R[1]:=R[2]+R[1];
R[0]:=R[1]xR[0];
```

Le reste de la question se traite par une assertion du même type que pour la question 6 (remplacer χ par ρ).

Q22. La valeur maximum de ρ s'obtient sur les arbres binaires parfaits (dont toutes les feuilles sont à la même profondeur) ; la valeur minimale sur les arbres filiformes. D'où les valeurs extrêmales : 1 et $\lceil \log_2 (n+1) \rceil$. Toutes les valeurs intermédiaires de ρ sont prises comme on s'en aperçoit en "greffant" un arbre parfait sur une feuille d'un arbre filiforme.

Q23. Comme les valeurs extrêmes de ρ sur $B_n(\Omega,X)$ sont 1 et $\lceil \log_2 (n+1) \rceil$, la moyenne de ρ est comprise entre ces bornes. (On pourrait aussi utiliser une forme de l'espérance analogue à celle de la question 13). D'où :

$$\lim_{n\to\infty} \frac{\overline{\rho n}}{\overline{\chi}_n} = 0 \ !$$

Q24. Comparaison des procédures *genere* et *genere2*

1. Temps de calcul, *genere(a)* procède en une seule passe sur l'arbre d'expression *a*, d'où un temps de calcul en $O(n)$. *genere2 (a)* procède en une passe, mais en chaque sommet visité est lancé un appel à la procédure *rho* ; le temps cumulé des appels de la procédure *rho* est proportionnel à la somme des tailles de tous les sous-arbres de l'arbre *a*. Cette quantité est d'ordre $O(n^2)$

2. Nombre de registres du programme généré. Il vaut $\sqrt{n}$ en moyenne pour *genere* et $O(\log_2 n)$ pour *genere2*. Dans le cas le pire, on a respectivement $O(n)$ et $O(\log_2 n)$.

Donc la procédure *genere* est plus rapide à l'exécution, mais conduit à une moins bonne utilisation des registres. On observe alors qu'on peut définir une procédure *genere3* qui opère de la façon suivante :

> *genere3 (a)* construit d'abord un arbre *b* isomorphe à *a*, mais dont les sommets contiennent un champ supplémentaire *rhoval* ; par évaluation en ordre postfixe *genere3* calcule en une seule passe les valeurs de ρ associées à chaque sommet et les range dans les champs *rhoval* correspondants. Puis *genere3* procède comme *genere3* en utilisant en chaque sommet le contenu de *rhoval* au lieu de lancer un appel à la procédure *rho*.

La procédure *genere3* opère en temps linéaire (mais en deux passes) tout en construisant un programme élémentaire d'au plus $\log_2 (n+1)$ registres.

4. Indications bibliographiques

Les procédures *genere* et *genere3* (amélioration décrite à la question Q24. de *genere2*) sont utilisées concurremment dans les compilateurs. Pour une description des problèmes d'allocation, on pourra consulter :

> R. Sethi, J. Ullman : The generation of optimal code for arithmetic expressions, *Journal of the Association for Computing Machinery* (J.A.C.M.), 17 (1970), pp. 715-728.

L'article original d'Ershov (Comm. ACM 1958) ne présente plus de nos jours que l'intérêt historique de montrer la difficulté de spécification d'algorithme à une période où les concepts d'arbres, structures de données et récursivité étaient encore mal dégagés.

Les parties II et III sont fondées sur l'article suivant :

> N.G. De Bruijn, D.E. Knuth, S.O. Rice : The average height of planted plane trees, in *Graph Theory and Computing*, R.C. Read Ed. Academic Press (1972), pp. 15-22.

La méthode d'Ershov est analysée en détail dans :

> P. Flajolet, J.C. Raoult, J. Vuillemin : The number of registers required for evaluating arithmetical expressions, *Theoretical Computer Science, 9* (1979), pp. 99-125.
>
> R. Kemp : The average number of registers to evaluate a binary tree optimally, *Acta Informatica,* 11 (1979), pp. 363-372.

Le premier de ces articles utilise des résultats de Delange ; le second comme l'article de De Bruijn *et al.* utilise la transformation de Mellin (voir la Q19).