RAIRO. INFORMATIQUE THÉORIQUE

JEAN-LUC RÉMY

Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire

RAIRO. Informatique théorique, tome 19, n° 2 (1985), p. 179-195

<http://www.numdam.org/item?id=ITA_1985__19_2_179_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 19, n° 2, 1985, p. 179 à 195)

UN PROCÉDÉ ITÉRATIF DE DÉNOMBREMENT D'ARBRES BINAIRES ET SON APPLICATION A LEUR GÉNÉRATION ALÉATOIRE (*)

par Jean-Luc RÉMY (1)

Communiqué par R. CORI

Résumé. — *L'article présente une méthode de type géométrique pour dénombrer les* arbres binaires *de taille n; on s'intéresse d'abord à un type d'objets plus complexes, les arbres binaires à feuilles numérotées, et on obtient pour ceux-ci un procédé de construction itératif très simple, l'insertion des feuilles par numéros croissants. Le nombre B'_n d'arbres binaires à feuilles numérotées vérifie la relation de récurrence $B'_n = 2 \times 2n-1 \times B'_{n-1}$. Pour chaque arbre de taille n, il y a évidemment $(n+1)!$ manières de numéroter ses feuilles. Ainsi le nombre B_n d'arbres binaires de taille n vérifie la relation de récurrence :*

$$B_n = 2.(2n-1)\,B_{n-1}/(n+1).$$

On dérive également du procédé précédent une technique de génération d'arbres binaires aléatoires suivant une loi uniforme. Cette technique procède en temps linéaire contrairement à celles de Knott et Rotem que l'on présente brièvement.

Abstract. — *We present a geometric method in order to enumerate binary trees with n nodes, investigating the class of binary trees with labeled leaves. A very simple device allows to generate all those trees by inserting the leaves iteratively, following the order given by the labels. It is then easy to verify that the number B'_n of binary trees with $(n+1)$ labeled leaves satisfies the relation $B'_n = 2.(2n-1)\,B'_{n-1}$. As there are $(n+1)!$ different possible labelings for the leaves, the number of binary trees satisfies:*

$$B_n = 2.(2n-1)\,B_{n-1}/(n+1),$$

which gives the classical formula and B_n is the Catalan number.

From this construction, we obtain also an efficient procedure for the random generation of a binary tree. This technique runs in linear time and is thus better than those of Knott and Rotem which are briefly recalled.

(*) Reçu en décembre 1983, révisé en septembre 1984.
(1) C.R.I.N., B.P. n° 239, 54506 Vandœuvre-lès-Nancy-Cedex, France.

0. INTRODUCTION

Les arbres binaires sont au cœur de l'algorithmique et de la combinatoire; ils fournissent par exemple des implantations efficaces pour des structures de données variées (dictionnaires, files de priorité).

On pourra lire en particulier Aho, Hopcroft et Ullman [AHU 74] (chap. 4), Horowitz et Sahni [HS 78] et pour des exemples plus particuliers Rémy [REM 80], Françon, Viennot et Vuillemin [FVV 78].

Il est donc important de bien savoir énumérer, dénombrer et engendrer de façon aléatoire cette catégorie d'objets. Le nombre B_n d'arbres binaires de taille n (c'est-à-dire avec n nœuds internes), ou n-ième nombre de Catalan, est en fait lié au dénombrement d'une famille importante d'objets. Il est ainsi possible de dénombrer les forêts, les parenthésages d'une expression mais aussi les séquences liées au problème du scrutin de ballotage [ROT 75] [COM 70], le nombre de chemins sous-diagonaux entre l'origine et le point de coordonnées (n, n) dans un treillis.

La méthode la plus couramment employée utilise la technique des fonctions génératrices. Schématiquement, il s'agit de calculer la fonction :

$$B(x) = \sum_{n \geq 0} B_n x^n.$$

Il est possible de façon générale de dériver de la définition structurelle d'une classe d'objets une équation vérifiée par la fonction génératrice associée [FLA 79]. Il reste alors à résoudre celle-ci et à calculer les coefficients B_n du développement de B. Dans le cas des arbres binaires, l'équation structurelle peut s'écrire :

$$\mathscr{B} = \square + \begin{array}{ccc} & \circ & \\ \mathscr{B} & & \mathscr{B} \end{array},$$

où $\square$ représente l'arbre vide.

La fonction génératrice $B(x)$ associée à $\mathscr{B}$ est définie par la relation :

$$B(x) = \sum_{A \in \mathscr{B}} x^{|A|},$$

où $|A|$ désigne la taille de A.

Pour passer de l'équation (1) à une équation vérifiée par B, il faut remplacer l'arbre vide $\square$ par 1 (puisque sa taille est 0) et remplacer $\begin{array}{ccc} & \circ & \\ \mathscr{B} & & \mathscr{B} \end{array}$ par le

produit $x B^2(x)$. En effet :

$$\sum_{A \in \overset{\circ}{\underset{B \quad B}{\wedge}}} x^{|A|} = \sum_{A_1, A_2 \in \mathcal{B}} x^{1+|A_1|+|A_2|} = x \cdot \sum_{A_1 \in \mathcal{B}} x^{|A_1|} \cdot \sum_{A_2 \in \mathcal{B}} x^{|A_2|}.$$

Ainsi $B(x)$ est solution de l'équation :

$$B(x) = 1 + x \cdot B^2(x). \tag{2}$$

Il s'agit d'une équation du second degré dont une seule solution est analytique à l'origine : $B(x) = (1 - \sqrt{1-4x})/2x$. Finalement, après calcul, on trouve l'expression :

$$B_n = \frac{1}{n+1}\binom{2n}{n}, \tag{3}$$

dans laquelle $\binom{n}{p}$ désigne le nombre de combinaisons de p éléments parmi n.

Nous proposons ici une technique de dénombrement moins classique mais beaucoup plus simple, fondée sur une manière de construire un arbre binaire : l'insertion des nœuds (ou ce qui revient au même des feuilles) à partir de l'arbre vide. Comme ce procédé n'est pas bijectif, on passe par l'intermédiaire d'une autre famille d'objets, les arbres binaires à feuilles numérotées. Nous montrons qu'il existe $2 \times (2n-1)$ façons d'insérer un nœud dans un arbre donné de taille $n-1$ et donc que :

$$B'_n = 2 \times 2n - 1 \times B_{n-1}$$

si B'_n désigne le nombre d'arbres à feuilles numérotées de taille n.

Comme il y a $(n+1)!$ manières de numéroter les $n+1$ feuilles d'un arbre de taille n, les nombres B_n et B'_n sont liés par la relation :

$$B'_n = (n+1)!\, B_n$$

et donc B_n vérifie la relation de récurrence :

$$B_n = \frac{2 \times (2n-1)}{(n+1)} B_{n-1}$$

de laquelle on déduit immédiatement

$$B_n = 2^n \cdot \frac{1.3.\ldots.2n-1}{1.2.\ldots.n} \times \frac{1}{n+1} = \frac{2n!}{n! \times n!} \cdot \frac{1}{n+1} = \binom{2n}{n} \cdot \frac{1}{n+1}.$$

Le procédé de construction introduit permet d'obtenir un arbre binaire suivant une loi de probabilité uniforme. Nous présentons au paragraphe 2 un algorithme de génération aléatoire très simple, linéaire et nécessitant pour chaque nœud ou feuille deux pointeurs vers les fils gauches et droits; nous transformons légèrement l'algorithme pour ne pas avoir à représenter les feuilles explicitement.

Au paragraphe 3, nous comparons notre algorithme avec ceux de Knott d'une part et de Rotem d'autre part. A la différence de ces derniers, nous ne sommes pas en mesure d'engendrer les arbres dans leur ordre naturel mais ceci est un problème relativement indépendant. En effet, il est assez simple de passer d'un arbre à son successeur immédiat. Il est possible à ce propos d'améliorer l'évaluation faite par Rotem en remarquant qu'en moyenne le nombre d'opérations pour passer d'un arbre à son successeur est indépendant de la taille de l'arbre. Cela s'explique par le fait qu'en général, la transition est très locale et que les cas de modifications radicales de la structure sont rares.

1. ARBRES BINAIRES A FEUILLES NUMÉROTÉES. DÉFINITIONS ET DÉNOMBREMENT

1.1. Définitions

DÉFINITION 1 : Un *arbre binaire* A est une famille finie non vide d'éléments du langage $\{g, d\}^*$ engendré par les symboles g, d, vérifiant les propriétés suivantes :

(1) $\forall x \in A, \forall y, z \in \{g, d\}^* \; x = yz \Rightarrow y \in A$ (hérédité);

(2) $\forall x \in A$: ou bien $xg \in A$ et $xd \in A$ (on dit que x est un *nœud* de A) ou bien $xg \notin A$ et $xd \notin A$ (on dit que x est une *feuille* de A).

La taille $|A|$ d'un arbre binaire A est le nombre de ses nœuds. La *racine* de A est le mot vide $\wedge$ (qui appartient à A par hérédité). On désigne par $\mathscr{B}$ l'ensemble des arbres binaires, par $\mathscr{B}_n$ l'ensemble de ceux de taille n et par B_n la cardinalité de $\mathscr{B}_n$.

PROPOSITION 1: *Il existe un arbre unique de taille* 0, *noté* $\square$. *D'autre part, si* A_1 *et* A_2 sont deux arbres binaires, notons $A_1 \otimes A_2$ *la famille* $\{\wedge\} \cup gA_1 \cup dA_2$. $A_1 \otimes A_2$ *est un arbre binaire tel que* $|A_1 \otimes A_2| = 1 + |A_1| + |A_2|$. *Réciproquement, pour tout arbre* A *de taille positive, soit :*

$$A_1 = \{y \in \{g, d\}^* \mid gy \in A\} \qquad \text{et} \qquad A_2 = \{y \in \{g, d\}^* \mid dy \in A\}.$$

Il est trivial que A_1 *et* A_2 *sont des arbres et que* $A = A_1 \otimes A_2$. $\square$

L'ensemble $\mathscr{B}$ des arbres binaires est donc l'unique solution dans $\mathscr{P}(\{g, d\}^*)$. de l'équation :

$$\mathscr{B} = \square + \mathscr{B} \otimes \mathscr{B}.$$

PROPOSITION 2 : *Dans un arbre de taille n, il y a* $n+1$ *feuilles.*

Démonstration : $\square$ *a* exactement une feuille. Si A_1, A_2 sont des arbres de taille n_1, n_2, par récurrence ils ont n_1+1 et n_2+2 feuilles.

Le nombre de feuilles de $A_1 \otimes A_2$ est égal à :

$$(n_1+1)+(n_2+1) = (n_1+n_2+1)+1 = |A_1 \otimes A_2|+1.$$

Représentation graphique : On définit sur un arbre A la relation *père de* :

$$\forall x \in A, \quad \forall y \in A, \; x \text{ père de } y \iff y = xg \quad \text{ou} \quad y = xd.$$

On note les feuilles de l'arbre par $\square$ et les nœuds par $\bigcirc$. Ainsi l'arbre $A_1 \otimes A_2$ est représenté par le graphe 

.

DÉFINITION 2 (sous-arbres) : Étant donné un arbre binaire A et un nœud ou une feuille x de A, le sous-arbre $A[x]$ de A enraciné en x est l'ensemble $\{z \in \{g, d\}^* \mid xz \in A\}$. Il est trivial que cet ensemble est un arbre. Si A' est un arbre binaire, soit $A[A'/x]$ l'ensemble $A - x.A[x] + x.A'$. C'est l'arbre obtenu par substitution de A' à x dans l'arbre A.

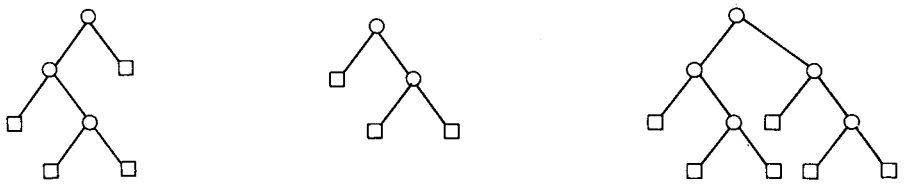
Exemple 1 : Soit :

$$A = \{\wedge, g, d, gg, gd, gdg, gdd\}, \qquad x = d$$

et :

$$A' = \{\wedge, g, d, dg, dd\}.$$

A, A' et $A[A'/x]$ admettent les représentations suivantes :



DÉFINITION 3 (arbres binaires à feuilles numérotées) : Un arbre binaire à feuilles numérotées A' est la donnée d'un arbre binaire A et d'une bijection ν

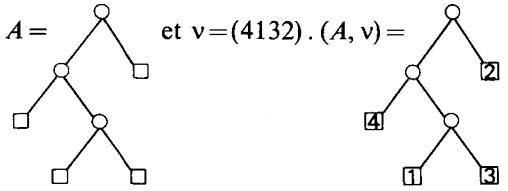
de l'ensemble des feuilles de A sur l'intervalle $1 \ldots |A|+1$. ν est appelé une numérotation des feuilles de A. Le nombre B'_n d'arbres binaires à feuilles numérotées de taille n est égal à $(n+1)! \times B_n$.

Représentation graphique: Nous représenterons une numérotation ν par la suite des numéros des feuilles prises en ordre symétrique.

On représente graphiquement un arbre binaire à feuilles numérotées en étiquetant les feuilles.

Exemple 2:

Soit :

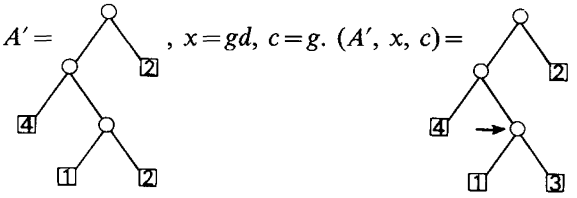


DÉFINITION 4 (arbres binaires pointés): Un arbre binaire pointé $A^\bullet$ est la donnée d'un arbre binaire à feuilles numérotées A', d'un nœud ou d'une feuille x de cet arbre et d'une marque $c \in \{g, d\}$. Le nombre $B^\bullet_n$ d'arbres binaires pointés de taille n est égal à $2 * (2n+1) * B'_n$.

Représentation graphique: On représente un arbre pointé en plaçant à côté du nœud désigné une flèche dirigée vers lui, à gauche si la marque est g, à droite sinon.

Exemple 3 :

Si :



L'objectif du paragraphe suivant est d'établir une bijection entre les arbres binaires pointés de taille n et les arbres binaires à feuilles numérotées de taille $n+1$.

1.2. Construction et décomposition d'un arbre à feuilles numérotées

Considérons un arbre binaire A' à feuilles numérotées de taille $n>0$. Soit f sa feuille de numéro $n+1$, p le père de cette feuille et x l'autre nœud ou feuille ayant p pour père (cf. *fig.* 1).

Supprimons de A' le nœud p et la feuille f. Précisément, soit A'' l'arbre obtenu en substituant à p le sous-arbre de A' enraciné par x (A'' est un arbre

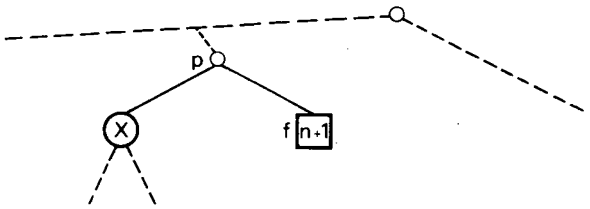
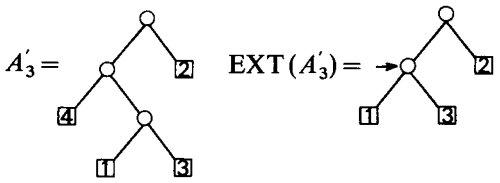


Figure 1.

à feuilles numérotées de taille $n-1$). Soit $\mathrm{EXT}(A')$ l'arbre pointé obtenu en associant à A'' le nœud x et une marque égale à d si f est à droite de x, égale à g sinon.

Exemple 4 :

Soit :



NOTATION : *On définit sur* $\{g, d\}$ *l'application* $^-$ *par* $\bar{g}=d$, $\bar{d}=g$.

Les deux définitions suivantes formalisent le procédé que nous venons de décrire.

DÉFINITION 5 : Soit $x \in \{g, d\}^* - \wedge$ un nœud ou une feuille. La *décomposition* de x est l'unique couple $(p, c) \in \{g, d\}^* \times \{g, d\}$ tel que $x = p.c$.

DÉFINITION 6 : Soit A' un arbre à feuilles numérotées de taille n strictement positive, f sa feuille de numéro $n+1$, (p, c) la décomposition de f et $x = p.\bar{c}$ le frère de f. *L'arbre extrait de* A' est l'arbre pointé $\mathrm{EXT}(A')$ défini par :

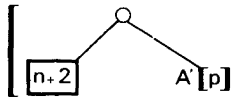
$$\mathrm{EXT}(A') = (A'[A'[x]/p], p, c).$$

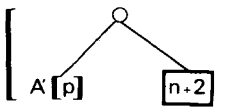
Vérifions que l'application EXT est une bijection de l'ensemble des arbres binaires numérotés non vides sur l'ensemble des arbres binaires pointés et définissons pour cela une opération réciproque INS.

DÉFINITION 7 : Soit $A^\bullet = (A', p, c)$ un arbre pointé. *L'arbre déduit de* $A^\bullet$ *par insertion d'une nouvelle feuille* est l'arbre à feuilles numérotées noté $\mathrm{INS}(A^\bullet)$ défini de la manière suivante.

Soit n la taille de $A^{\bullet}$. Alors:

$\mathrm{INS}(A^{\bullet}) =$

cas $c = g$ *alors* A' $\left[\right.$  $/p \left.\right]$

$c = d$ *alors* A' $\left[\right.$  $/p \left.\right]$

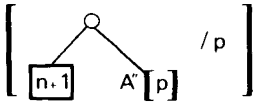
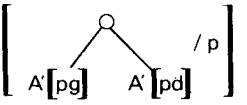
PROPOSITION 3: INS *et* EXT *sont deux applications réciproques.*

Démonstration: (1) Soit A' un arbre à feuilles numérotées de taille n strictement positive.

Supposons, sans perdre de généralité, que la feuille de numéro $n+1$ de A' s'écrive pg. Soit:

$$A'' = A'[A'[pd]/p]; \qquad \mathrm{EXT}[A'] = (A'', p, g)$$

et :

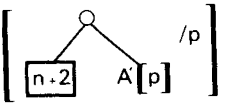
$\mathrm{INS}(\mathrm{EXT}(A')) = A''$ $\left[\right.$  $/p \left.\right]$ $=$ A'' $\left[\right.$  $/p \left.\right]$

$$= A'[A'[pd]/p][A'[p]/p]$$

$$= A'[A'[p]/p] = A'.$$

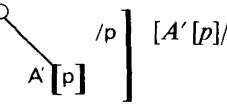
(2) Réciproquement, soit $A^{\bullet} = (A', p, c)$ un arbre pointé de taille $n \geqq 0$.

Supposons $c = g$. Soit :

$$A'' = \mathrm{INS}(A^{\bullet}) = A' \left[\right. \text{  } /p \left.\right] ; \qquad A''[pd] = A'[p].$$

Donc :

$$\mathrm{EXT}[\mathrm{INS}[A^{\bullet}]] = (A''[A''[pd]/p], p, g)$$

$$= (A' \left[\right. \text{  } /p \left.\right] [A'[p]/p], p, g)$$

$$= (A'[A'[p]/p], p, g)$$

$$= (A', p, g).$$

1.3. Dénombrement des arbres binaires

PROPOSITION 4:

(*a*) $B'_0 = 1$;

(*b*) $B'_{n+1} = B'_n \times 2 \times (2n+1)$;

(*c*) $B'_n = 2^n . (2n-1)!!$ [où $(2n-1)!! = 1.3.5.7.. (2n-1)$].

Démonstration:

(*a*) Le seul arbre de taille zéro est 1 ;

(*b*) Résulte de ce que INS est une bijection entre $\mathscr{B}_n$ et $\mathscr{B}'_{n+1}$;

(*c*) Résulte de (*a*) et (*b*).

PROPOSITION 5:

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

Démonstration: Par la proposition 3,

$$B_n = \frac{1}{(n+1)!} B'_n = \frac{2^n . 1 \times 3 \times \ldots \times (2n-1)}{n!(n+1)}$$

$$= \frac{1}{n+1} \cdot \frac{2^n n! \, 1 \times 3 \times \ldots \times (2n-1)}{n! n!}$$

$$= \frac{1}{n+1} \cdot \frac{(2n)!}{n! n!} = \frac{1}{n+1}\binom{2n}{n}.$$

2. GÉNÉRATION ALÉATOIRE D'ARBRES BINAIRES SELON UNE LOI UNIFORME

L'opération d'insertion définie au paragraphe 1.2 permet d'écrire une procédure Genere de génération aléatoire d'arbres binaires selon une loi uniforme. Soit Efface l'application de $\mathscr{B}'$ dans $\mathscr{B}$ qui associe à un arbre à feuilles numérotées l'arbre sous-jacent. Tout arbre à feuille numérotée s'écrit de manière unique:

$$\mathrm{INS}^n(\square, x_1, c_1, \ldots, x_n, c_n),$$

où $c_i \in \{g, d\}$ et $x_i \in \mathrm{INS}^{i-1}(\square, x_1, c_1, \ldots, x_{i-1}, c_{i-1})$. Engendrer un arbre de taille n se résume donc à choisir aléatoirement et indépendamment les valeurs $c_1, \ldots, c_n$ dans $\{g, d\}$ et à choisir aléatoirement et indépendamment les nœuds ou feuilles $x_1, \ldots, x_n$ puis à effacer les numéros placés aux feuilles.

Pour effectuer ces choix, on se donne deux procédures de tirage au hasard :

Hasard 1 ( ) qui rend g ou d avec les probabilités égales,

Hasard 2 (A) qui, étant donné une partie finie A de $\{g, d\}^*$, rend un élément de A avec une probabilité 1/card(A).

L'insertion utilise une procédure Insere (x, c, m) avec une variable globale A de type arbre($m = |A| + 1$). Insere (x, c, m) transforme A en INS(A, x, c).

Procédure Genere (n, A)

$A := \square$
pour m *de* 1 *jusqu'à* n *faire*
  début
    $c :=$ Hasard 1 ( )
    $x :=$ Hasard 2 (A)
    Insere (x, c, m)
  fin

Une première représentation des arbres consiste à associer bijectivement à chaque nœud ou feuille x une adresse $\alpha(x)$ comprise entre 1 et $2n+1$ et à représenter les liens gauche et droit par deux tableaux GAU$[1 \ldots 2n+1]$ et DRO$[1 \ldots 2n+1]$ à valeurs dans $[0 \ldots 2n+1]$ de telle manière que l'on ait les relations suivantes (cf. *fig.* 2) :

$$\alpha(\wedge) = 1 \qquad \text{et} \qquad \text{si } \alpha(x) = i.$$

$$\text{GAU}[i] = \begin{cases} \alpha(xg) & \text{si } x \text{ est un nœud} \\ 0 & \text{sinon} \end{cases}$$

$$\text{DRO}[i] = \begin{cases} \alpha(xd) & \text{si } x \text{ est un nœud} \\ 0 & \text{sinon} \end{cases}$$

L'affectation $A := \square$ se traduit par GAU $[1] :=$ DRO$[1] := 0$. Le tirage de x peut être représenté par le tirage d'un nombre i entre 1 et $2m-1$, puisqu'au début de la m-ième itération, $|A| = m-1$ et donc qu'il y a en tout $2m-1$ nœuds ou feuilles.

La procédure INSERE consiste à :

— donner des adresses ($2m$ et $2m+1$) au nœud et à la feuille insérée, et affecter les lignes correspondantes des tableaux GAU et DRO;

— modifier les valeurs de GAU [i] et DRO [i].

Les textes détaillés des procédures Insere et Genere sont donnés dans [REM 80*b*].

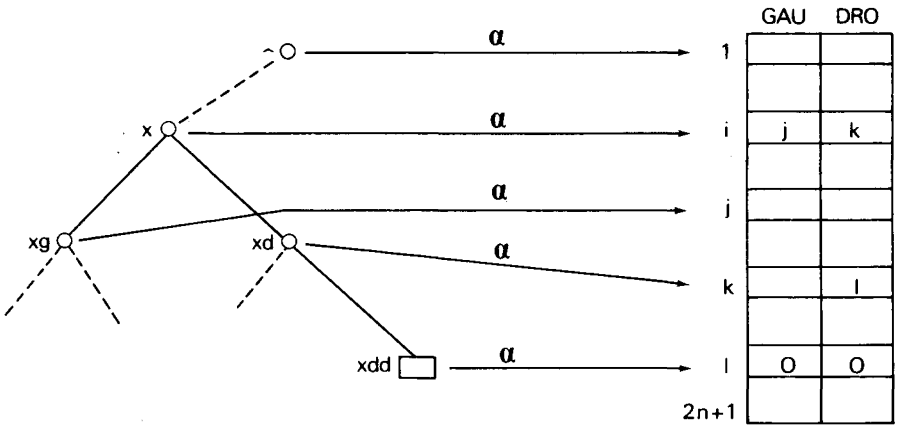


Figure 2. — Une première représentation.

Évaluation de la complexité de la procédure

Cette première procédure nécessite deux tableaux de dimension $2n+1$. D'autre part le coût de chaque insertion est de six affectations, plus deux affectations à c et i. D'où un coût global de $8 \times n$ affectations. Il est aussi possible de compactifier les deux tableaux en un seul et de n'utiliser ainsi que $2n+1$ emplacements [REM 80 *b*].

3. COMPARAISON AVEC D'AUTRES ALGORITHMES

Nous comparons notre algorithme avec deux autres proposés respectivement par Knott [KNO 76] et Rotem et Varol [RV 78]. Ces deux algorithmes sont de complexité supérieure [temps d'exécution en $O(n \log n)$ et occupation mémoire en $O(n^2)$]. Le deuxième algorithme est une occasion de présenter des correspondances intéressantes entre les arbres binaires et d'autres structures de données. D'autres techniques, que nous ne présenterons pas ici, peuvent être trouvées dans [LIU 80], [PRO 80], [SF 80].

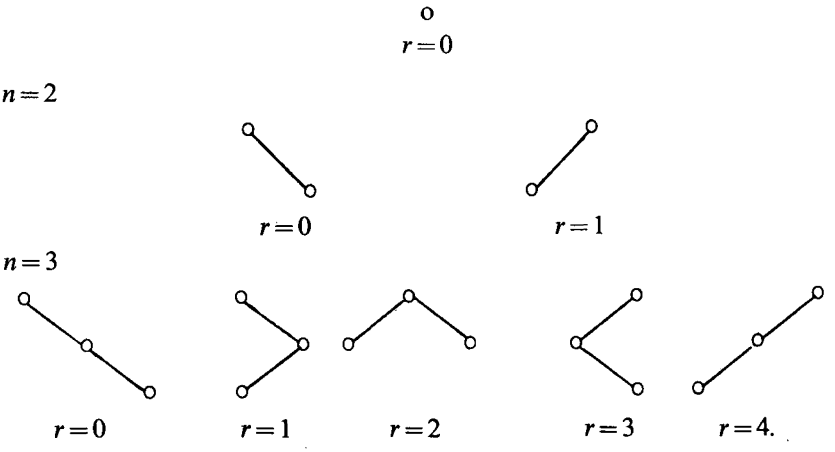
3.1. Algorithme de Knott

Knott utilise l'ordre lexicographique sur les arbres binaires pour définir une numérotation de ceux-ci. Engendrer un arbre aléatoirement consiste à choisir un nombre au hasard entre O et B_{n-1} et à utiliser une procédure décode pour déterminer l'arbre dont le numéro est le nombre choisi. Présentons successivement l'ordre lexicographique, la procédure rang de numérotation et la procédure décode.

L'ordre lexicographique sur les arbres binaires à N nœuds compare d'abord le nombre de nœuds du sous-arbre gauche, en cas d'égalité, compare les sous-arbres gauches eux-mêmes et en dernier ressort compare les sous-arbres droits.

Illustrons cette définition en énumérant, en ordre lexicographique, les arbres à un, deux puis trois nœuds; pour simplifier la représentation, nous n'avons pas fait figurer les feuilles sur les arbres en question.

$n=1$

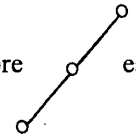


Le rang d'un arbre possédant n nœuds au total dont p nœuds à gauche de sa racine peut être calculé en additionnant :

— le nombre d'arbres possédant n nœuds au total dont moins de p nœuds à gauche, noté dans la suite $B'_{n,p}$;

— le rang, parmi les arbres à p nœuds, du sous-arbre de gauche, à multiplier par le nombre total d'arbres de $n-1-p$ nœuds;

— et finalement le rang, parmi les arbres de $n-1-p$ nœuds, du sous-arbre de droite.

Par exemple le rang de l'arbre  est égal à $3+1+0$.

Les nombres $B'_{n,p}$, pour $n=0, 1, \ldots, N$ et $p=1, \ldots, n-1$, peuvent être calculés une fois pour toutes et rangés dans un tableau triangulaire occupant approximativement $N^2/2$ emplacements.

Ce tableau constitué, il est possible de déterminer un arbre connaissant le nombre N de ses nœuds et son rang r compris entre 0 et $B_N - 1$.

(1) La taille p du sous-arbre gauche est donnée par le plus grand nombre $B'_{N,p}$ inférieur ou égal à r.

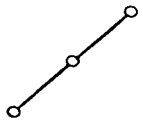
(2) Soit $q = n - 1 - p$ et $r' = r - B'_{N,p}$. Le quotient et le reste de r' par B_q sont respectivement égaux aux rangs du sous-arbre gauche et du sous-arbre droit de l'arbre recherché.

(3) Il reste alors à appliquer récursivement l'algorithme pour déterminer ces sous-arbres.

Exemple : $N = 3$, $r = 4$:

(1) $B'_{3,1} = 2$, $B'_{3,2} = 3$, soit $p = 2$, $q = 0$, $r' = 1$, $B_p = 1$.

(2) Le sous-arbre gauche a pour rang 1; le sous-arbre droit est vide, d'où l'arbre :



Chaque appel de l'algorithme récursif permet de placer un nœud, il nécessite une recherche dichotomique dans une ligne du tableau B', de coût algorithmique, et des opérations arithmétiques de coût négligeable.

Le coût temporel total est $N \log N$. Le coût en espace est proportionnel à N^2.

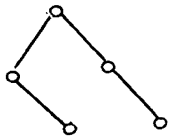
3.2. Algorithme de Rotem et Varol

L'idée de Rotem et Varol est très sensiblement différente. Elle consiste à associer à un arbre binaire un chemin joignant deux points opposés d'un carré de côté N en restant toujours en dessous de la diagonale puis à calculer une fonction de numérotation de ces chemins. Pour construire le chemin sous-diagonal, les auteurs passent par l'intermédiaire d'un parenthésage de l'arbre.

Nous ne donnerons qu'une idée intuitive de l'algorithme. Plus de détails sont donnés dans [RV 78] et [REM 80 *b*].

Construction du chemin sous-diagonal

Considérons l'arbre suivant (où les feuilles ne sont pas représentées)

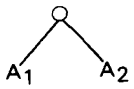


Parcourons cet arbre en ordre préfixé et notons par des parenthèses ouvrantes et fermantes la descente sur un fils aîné et le passage au frère cadet ou à l'oncle le plus proche. L'absence d'un nœud sur une branche se traduit par des parenthèses ouvrante et fermante consécutives. On obtient ici :

$$(\ (\ )\ (\ )\ )\ (\ )\ (\ ).$$

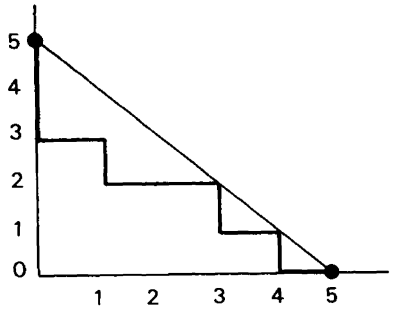
Une expression formelle de la fonction de parenthésage est donnée par les équations récursives :

Par $(\square) = \wedge$.

Par :  $= (\text{Par}\ (A_1))\ \text{Par}\ (A_2)$.

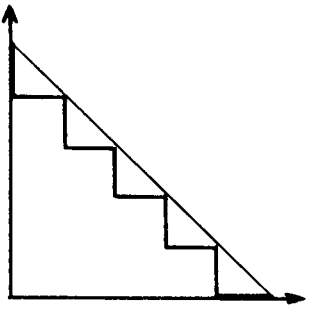
Utilisons maintenant cette séquence pour décrire un chemin sous-diagonal. Celui-ci part du point de coordonnées (O, N). Chaque parenthèse ouvrante ou fermante est codée par un segment unitaire vertical ou horizontal. Le chemin aboutit naturellement au point (N, O). Nous ne décrivons pas ici le décodage inverse.

Pour le même exemple, on obtient le chemin suivant :



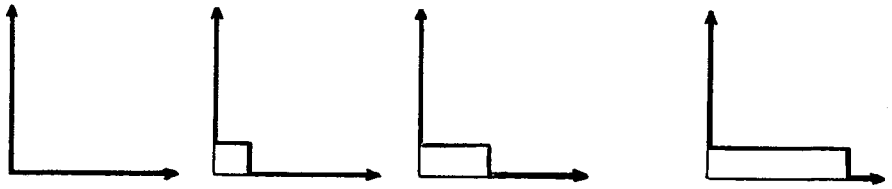
Numérotation des chemins

Nous classons les chemins selon un ordre lexicographique, mais en remontant ceux-ci depuis leur extrémité.

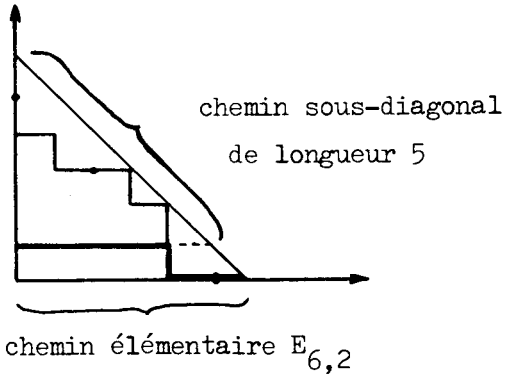


Voici le chemin de côté 5 de plus haut rang :

Les chemins les plus significatifs sont les chemins élémentaires suivants :



Il est possible de décomposer un chemin sous-diagonal en un chemin élémentaire, le plus bas possible, plus un nouveau chemin sous-diagonal. La figure suivante présente la décomposition d'un chemin de longueur 6.



La décomposition présente la propriété remarquable que le rang du chemin total est la somme du rang du chemin élémentaire et de celui du sous-chemin obtenu. De nouveau, les rangs des chemins élémentaires peuvent être calculés une fois pour toutes et rangés dans un tableau triangulaire occupant la même

place que celui de Knott. La procédure de décodage associée est récursive et utilise à chaque appel une recherche dichotomique du rang du chemin élémentaire. Illustrons-la simplement sur un exemple.

n \ k	1	2	3	4	5	6
1.....	1					
2.....	1	2				
3.....	1	3	5			
4.....	1	4	9	14		
5.....	1	5	14	28	42	
6.....	1	6	20	48	90	132

Rangs des chemins élémentaires

$r=86$

ligne 6 :

$$86=48+38, \qquad k=4;$$

ligne 5 :

$$38=28+10, \qquad k=4;$$

ligne 4 :

$$10=9+1, \qquad k=3;$$

ligne 3 :

$$1=1+0, \qquad k=1.$$

On retrouve le chemin illustrant le procédé de décomposition.

Complexité de la procédure

L'occupation mémoire est équivalente à celle de la procédure de Knott. Le temps d'exécution est du même ordre de grandeur, bien que les opérations à faire soient plus simples et que l'on obtienne plus directement l'arbre associé grâce à sa forme parenthésée. Cette procédure, comme la précédente, est donc plus coûteuse que celle que nous proposons.

REMERCIEMENTS

Je tiens à remercier très chaleureusement Jean Vuillemin et Philippe Flajolet à qui j'ai présenté pour la première fois, il y a longtemps, cette méthode de dénombrement; le premier m'a suggéré de l'appliquer à la génération d'arbres. Ce texte n'aurait pas vu le jour également sans l'insistance et les encouragements de Pierre Lescanne, Jean-Pierre Jouannaud et Jean-Marc Steyaert qui en

ont relu une première version. Enfin j'ai été accompagné dans l'ultime et tardive rédaction par Jocelyne Bazin et Christiane Floc'h. Je les remercie très sincèrement.

BIBLIOGRAPHIE

[AHU 74] A. V. AHO, J. E. HOPCROFT et J. D. ULLMAN, *The Design and Analysis of Algorithms*, Addison-Wesley, Reading, Mass, 1974.

[COM 70] L. COMTET, *Analyse combinatoire*, vol. 1, 2, Presses Universitaires de France, Paris, 1970.

[FLA 79] P. FLAJOLET, *Analyse d'algorithmes de manipulation d'arbres et de fichiers*, Thèse, Université de Paris-Sud, Paris, 1979.

[FRA 79] J. FRANÇON, *Combinatoire des structures de données*, Thèse, Faculté des Sciences de Strasbourg, 1979.

[FVV 78] J. FRANÇON, G. VIENNOT et J. VUILLEMIN, Description and Analysis of an Efficient Priority Queue Representation, Proc. of 19th I.E.E.E. Symp. on Found. of Comp. Sc., 1979.

[HV 78] E. HOROWITZ et S. SAHNI, *Fundamentals of Computer Algorithms*, Computer Science Press, Potomac, Maryland, 1978.

[KNO 77] G. D. KNOTT, *A Numbering System for Binary Trees*, Comm. of A.C.M., vol. 20, n° 2, 1977, p. 113-115.

[LIU 80] C. L. LIU, *Generation of k-ary trees* in *Actes du 5^e coll. de Lille sur les Arbres en Algèbre et en Programmation*, Lille, 1980, p. 45-53; également Rapport n° 27, I.N.R.I.A., Rocquencourt, 1980.

[PRO 80] A. PROSKUROWSKI, *On the Generation of Binary Trees*, J. A.C.M., vol. 27, n° 1, 1980, p. 1-2.

[REM 80] J. L. RÉMY, *Construction, évaluation et amélioration systématiques de structures de données*, R.A.I.R.O. Informatique théorique, vol. 14, n° 1, 1980, p. 83-118.

[REM 80 *b*] J. L. RÉMY, *Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire*, Actes des 3^es Journées de la RCP Complexité, Nice 1980 *et* Rapport 80-P-053, C.R.I.N. 1980.

[ROT 75] D. ROTEM, *On a Correspondence between Binary Trees and a Certain Type of Permutation*, Information Processing Letters, vol. 4, n° 1, 1975, p. 58-61.

[RV 78] D. ROTEM et Y. L. VAROL, *Generation of Binary Trees from Ballot Sequences*, J. A.C.M., vol. 25, n° 3, 1978, p. 396-404.

[SF 80] M. SOLOMON et R. A. FINKEL, *A Note on Enumerating Binary Trees*, J. A.C.M., vol. 27, n° 1, 1980, p. 3-5.