ELSEVIER

Information Processing Letters 64 (1997) 305-308

Information Processing Letters

Uniform generation of a Schröder tree

L. Alonso [a], J.L. Rémy [a,b], R. Schott [b,*]

[a] INRIA-Lorraine, CRIN-CNRS, Université de Nancy 1, 54506 Vandœuvre-lès-Nancy, France

[b] CRIN-CNRS, Université de Nancy 1, 54506 Vandœuvre-lès-Nancy, France

Received 24 June 1997; revised 26 August 1997

Communicated by D. Gries

Abstract

We present a simple $O(n)$ algorithm that generates uniformly a Schröder tree of size n. The basic idea is to choose a slightly enlarged probability space where uniformity can be achieved. © 1997 Elsevier Science B.V.

Keywords: Schröder tree; Linear time; Generation; Rejection; Uniform; Computational complexity; Algorithms

1. Introduction

Random generation of tree structures is now a very active research area with motivations coming from statistical complexity analysis, graphics, percolation theory, etc. This paper is a continuation of [1-3], where we address the problem of generating with uniform probability one tree among a set of trees whose cardinality is either exactly known by a close formula or given by a combinatorial sum. The family of trees under consideration was investigated by Schröder in connection with bracketing problems [8]. Among the recent works on Schröder trees, we would like to mention [5,6,10,11].

Our algorithm builds uniformly a random Schröder tree of size n. This algorithm is simple, easy to code, and has an average complexity in $O(n)$. The algorithm has two main parts. In the first part, we choose the number k of leaves with some appropriate probability. In the second part, we build a random Schröder tree that has k leaves and n nodes: this construction can be done in linear time [2].

2. Schröder trees: Definitions and basic facts

We recall the necessary definitions and properties and refer to [5,10] for more details about these trees.

A Schröder tree T is either the tree consisting of a root alone, $T = r$, or an ordered tuple $[r, T_1, \ldots, T_l]$, where $l \geqslant 2$ and $T_1, \ldots, T_l$ are smaller Schröder trees. The first symbol r is called the root of T and the roots

* Corresponding author.

PII S0020-0190(97)00174-9

of $T_1, \dots, T_l$ are called the sons of T. A son-less node is called a leaf. The number of Schröder trees with m leaves is denoted by $s(m)$. Recall the linear recurrence

$$3(2m-1)s(m) = (m+1)s(m+1) + (m-2)s(m-1), \quad m \geqslant 2,$$

$$s(1) = s(2) = 1,$$

and the generating function

$$\sum_{m \geqslant 1} s(m)x^m = \frac{1}{4}(1 + x - \sqrt{1 - 6x + x^2}).$$

The first values of the $s(m)$s are as follows:

$$s[1..10] = (1, 1, 3, 11, 45, 197, 903, 4279, 20793, 103049).$$

More precisely, the number of Schröder trees with n nodes and k internal nodes is given by [2,4]:

$$S_{n,k} = \frac{1}{n}\binom{n}{k}\binom{n-1-2k+k-1}{n-1-2k},$$

where $1 \leqslant k \leqslant (n-1)/2$ and $s(n) = \sum_{k=1}^{n-1} S_{n+k,k}$.

3. Generation algorithm

Using the formula given above, we can easily show that

$$S_{n,k} = \left(\frac{n-2}{n}\right)\left(\frac{n}{n-k}\right)\left(\frac{n-1}{n-k-1}\right)A_{n-2,k},$$

where n is the number of internal nodes and $A_{n-2,k} = (n-3)!/(k!(k-1)!(n-1-2k)!)$ is the number of unary-binary trees that have $n-2$ nodes and $k-1$ binary nodes.

Now consider the ratio

$$S_{n,k}/\left(\sum_{k=1}^{(n-1)/2} S_{n,k}\right) = c_{n,k}/\left(\sum_{k=1}^{(n-1)/2} c_{n,k}\right)$$

with

$$c_{n,k} = \left(\frac{n}{2(n-k)}\right)\left(\frac{n-1}{2(n-k-1)}\right)A_{n-2,k}.$$

Our algorithm decomposes in two main steps:

Step 1. Choose a number k with probability $S_{n,k}/(\sum_{k=1}^{(n-1)/2} S_{n,k})$. This is done by applying two times the rejection technique [1].

Step 2. Generate uniformly a tree among the $S_{n,k}$. This is done in linear time using the method presented in [2].

Below, we give some details about these steps and their complexity analysis.

3.1. Step 1: Choice of the number k

The algorithm for choosing k (number of internal nodes) works as follows:

while (true)
{
generate k with probability $A_{n-2,k}/(\sum_{k=1}^{(n-1)/2} A_{n-2,k})$;
accept this choice with probability $(\frac{n}{2(n-k)})(\frac{n-1}{2(n-k-1)})$;
if this choice is accepted return k
}

Proposition 1. *This algorithm chooses a number k with probability*

$$S_{n,k}/\Big(\sum_{k=1}^{(n-1)/2} S_{n,k}\Big)$$

and the average complexity of this step is in $\mathrm{O}(n)$.

Proof. This is pretty clear. Indeed, for an iteration of the algorithm, we have the two following possibilities:
- Choose a number k with probability

$$\frac{(\frac{n}{2(n-k)})(\frac{n-1}{2(n-k-1)})A_{n-2,k}}{\sum_{k=1}^{(n-1)/2} A_{n-2,k}} = \frac{c_{n,k}}{\sum_{k=1}^{(n-1)/2} A_{n-2,k}}.$$

- Choose nothing, with probability $A = 1 - \sum c_{n,k}/\sum A_{n-2,k} < 1$.

Now, consider the sequence of events: k is chosen in the first step with probability $c_{n,k}/(\sum_{k=1}^{(n-1)/2} A_{n-2,k})$, in the second step with probability $Ac_{n,k}/(\sum_{k=1}^{(n-1)/2} A_{n-2,k})$, …, in the kth step with probability $A^{k-1}c_{n,k}/(\sum_{k=1}(n-1)/2A_{n-2,k})$.

Therefore, it is chosen with probability:

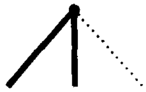
$$\frac{c_{n,k}}{\sum_{k=1}^{(n-1)/2} A_{n-2,k}}(1 + A + A^2 + \cdots) = \frac{c_{n,k}}{(1-A)\sum_{k=1}^{(n-1)/2} A_{n-2,k}} = \frac{c_{n,k}}{\sum_{k=1}^{(n-1)/2} c_{n,k}} = \frac{S_{n,k}}{\sum_{k=1}^{(n-1)/2} S_{n,k}}.$$

Moreover, we can see that the first iteration of the loop is done with probability 1, the second iteration with probability A, …. Therefore, the average number of times, the loop is iterated is $1 + A + A^2 + \cdots = 1/(1-A)$. Since the choice of k with probability $A_{n-2,k}/(\sum A_{n-2,k})$ has average complexity $\mathrm{O}(n)$ and the rejection step requires $\mathrm{O}(1)$ operations, the average complexity of the choice is $\mathrm{O}(n/(1-A))$.

But we have $A_{n,k}/4 \leqslant c_{n,k} \leqslant A_{n-2,k}$ when $1 \leqslant k \leqslant (n-1)/2$, therefore $1/(1-A) = \mathrm{O}(1)$. □

3.2. Step 2: Uniform generation of a tree among $S_{n,k}$

Using the algorithm of [2], the uniform generation of a tree among $S_{n,k}$, is done with the patterns (k, M_1) and $(n-k, \bullet)$ where



as follows:

- *Coding*: This procedure is similar to the one that is known for classical trees (see [2]). For example, binary trees are coded by Dyck words and unary binary trees by Motzkin words. Each Schröder tree is coded by a word of a language S whose construction is detailed in [2].
- *Generation*: The patterns M_1 and $\bullet$ are first mixed, then some edges are added to the resulting word, finally we apply the cycle lemma [4]; this gives a word of S that is in 1–1 correspondence with a Schröder tree.

This gives a random tree with k internal nodes that have at least two children and $n-k$ leaves, as required. The average time complexity of this step is in $\mathrm{O}(n)$.

References

[1] L. Alonso, Uniform generation of a Motzkin word, Theoret. Comput. Sci. 134 (1994) 529–536.
[2] L. Alonso, J.L. Rémy and R. Schott, A linear time algorithm for the generation of trees, Algorithmica 17 (1997) 162–182.
[3] L. Alonso and R. Schott, Random Generation of Trees, Kluwer Academic Publishers, Dordrecht, 1995.
[4] N. Dershowitz and S. Zaks, Patterns in trees, Discrete Appl. Math. 25 (1989) 241–255.
[5] D. Foata and D. Zeilberger, A classical proof of a recurrence for a classical sequence, Preprint.
[6] D. Gouyou-Beauchamps and B. Vauquelin, Deux propriétés combinatoires des nombres de Schröder, RAIRO Inform. Théor. 22 (3) (1988) 361–388.
[7] J.L. Rémy, Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire, RAIRO Inform. Théor. 19 (2) (1985) 179–195.
[8] E. Schröder, Vier combinatorische Probleme, Z. Math. Physik 15 (1870) 361–376.
[9] L.W. Shapiro and A.B. Stephens, Bootstrap percolation, the Schröder numbers and the n-kings problem, SIAM J. Discrete Math. 4 (1991) 275–280.
[10] R.P. Stanley, Hipparchus, Plutarch, Schröeder and Hough, Preprint.
[11] J. West, Generating trees and the Catalan and Schröder numbers, Discrete Math. 146 (1995) 247–262.