Information Processing Letters 41 (1992) 21–23
North-Holland 21 January 1992

Generating binary trees at random

M.D. Atkinson and J.-R. Sack

School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6

Communicated by S.G. Akl
Received 21 December 1990
Revised 17 September 1991

Abstract

Atkinson, M.D. and J.-R. Sack, Generating binary trees at random, Information Processing Letters 41 (1992) 21–23.

We give a new constructive proof of the Chung–Feller theorem. Our proof provides a new and simple linear-time algorithm for generating random binary trees on n nodes; the algorithm uses integers no larger than $2n$.

Keywords: Analysis of algorithms, binary trees, bracket sequences, data structures

1. Introduction

Methods for generating binary trees on n nodes have been considered by several authors (see [4,8] and [6] also for additional references). In most cases the focus has been on generating all binary trees in some order or on ranking and unranking them. The number of binary trees on n nodes is the Catalan number $\binom{2n}{n}/(n+1)$ which is exponential in n ($\approx 4^n$) and so these computations cannot be carried out very easily unless n is small. Unranking algorithms allow binary trees to be generated uniformly at random and this is often more useful than being able to list all the possible trees. Unfortunately, numbers which are exponential in n enter these calculations and this makes them impracticable unless n is small.

The problem of generating binary trees uniformly at random without introducing exponentially large numbers was overcome by Arnold and Sleep [1] and Martin and Orr [6]; they gave linear time algorithms which used integers of size O(n) for generating a random binary tree. We shall present a new and simpler solution having the same advantages. Our solution is based on a constructive version of the Chung–Feller theorem on coin-tossing, which has not previously been applied to this area. Our treatment also provides a new proof of the Chung–Feller theorem.

The set of binary trees on n nodes is well known to be in one-to-one correspondence with many other sets of combinatorial objects including rooted (ordered) trees with n branches, triangulations of a convex $(n+2)$-gon, lattice paths from $(0, 0)$ to (n, n) which do not cross the diagonal, and well-formed bracket sequences with n pairs of brackets. The one-to-one correspondences are explicit and efficient to compute in linear time; thus a uniform random generator for binary trees gives rise to a uniform random generator for all these other objects, and vice versa. We shall focus on generating well-formed sequences of brackets.

2. Terminology

We begin with some terminology. It is convenient to denote the left and right bracket symbols by λ and ρ respectively. Thus a bracket sequence (well formed or not) corresponds to a word over

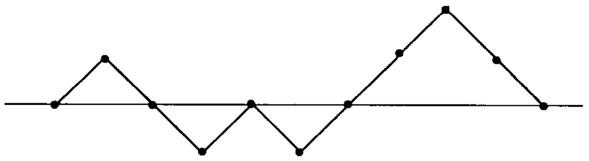

Fig. 1. The zigzag diagram corresponding to the word $\lambda\rho\rho\lambda\rho\lambda\lambda\lambda\rho\rho$.

the alphabet $\{\lambda, \rho\}$. A word such as $\lambda\rho\rho\lambda\rho\lambda\lambda\lambda\rho\rho$ may be pictured as a zigzag diagram drawn from some base line where each upward edge represents λ and each downward edge represents ρ (see Fig. 1).

A word is said to be *balanced* if it contains equal numbers of λ's and ρ's. Balanced words are precisely those whose diagram returns to the base line. A balanced word w is said to be *reducible* if it may be written $w = w_1 w_2$ with w_1, w_2 each balanced and nonempty, otherwise w is *irreducible*.

A balanced word is defined to have *defect i* if its diagram has precisely $2i$ links below its base line. Defect 0 words are called *well-formed* and corresponds to well-nested bracket sequences. Observe that the defect of a word is easily found by a summation: we scan the word of length k from left to right regarding each λ as $+1$, each ρ as -1, and computing the partial sums 0, $s_1, \ldots, s_k$; the final sum is zero and the number of negative interim sums s_j at odd indices j is the defect. We call this calculation *partial summation*.

For any word w let w^* denote the result of replacing all occurrences of λ by ρ and ρ by λ. The following two results are immediate consequences of these definitions.

Lemma 1. *If a balanced word w is irreducible, then one of w and w^* is well-formed; in fact, $w = \lambda u \rho$ where u is well-formed, or $w = \rho u \lambda$ where u^* is well-formed. If a balanced word w is well-formed, then $\lambda w \rho$ is irreducible.*

Lemma 2. *A balanced word w has a unique factorisation as $w = w_1 w_2 \cdots w_k$, where each w_i is irreducible. If w is well-formed, so is each w_i.*

3. The algorithm

Let B_n denote the set of $\binom{2n}{n}$ balanced words of length $2n$, and let B_{ni} denote the subset of balanced words of defect i. Clearly B_n is the disjoint union of $B_{n0}, B_{n1}, \ldots, B_{nn}$. The Chung–Feller Theorem, see [2, Theorem 2A] and [3, p.94], states that these subsets all have the same size. The central idea of our algorithm is to use a new constructive proof of this theorem which depends on explicit 1–1 correspondences between these sets.

Our algorithm has the following form:

Algorithm RANDOM BRACKET SEQUENCE

Input: An integer n.

Output: A well-formed word of length $2n$ over the alphabet $\{\lambda, \rho\}$.

1. Generate a uniformly random combination L of n integers from $\{1, 2, \ldots, 2n\}$
2. Define a random member $x = (x_1 x_2 \cdots x_{2n})$ of B_{2n} by the rule $x_i = \lambda$ if $i \in L$, $x_i = \rho$ if $i \notin L$.
3. Return the well-formed member of B_{2n} to which x corresponds.

Steps 1 and 2 of the algorithm are straightforward. To generate a combination of n integers from $\{1, 2, \ldots, 2n\}$ uniformly at random, and hence a member of B_n, we may use the technique described, for example, in [5, p.137] (see also [7, pp. 189–198] for more discussion on this topic). This technique takes only linear time and uses only integers less than $2n$.

To implement step 3 we need to define some suitable correspondences. We denote by $|w|$ the length of w and by card(S) the cardinality of a set S. We now define a map $\Phi_n : B_n \to B_{n0}$. The definition is inductive. For $n = 0$, we define Φ_0 in the only way possible: it maps the empty string to the empty string. For $n > 0$ and $w \in B_n$ we begin by expressing w as $w = uv$, where u is irreducible, $|u| = r > 0$, $|v| = s \geqslant 0$; then we define Φ_n by the rules

$$\Phi_n(w) = u\Phi_s(v) \quad \text{if } u \text{ is well-formed,}$$

$$\Phi_n(w) = \lambda\Phi_s(v)\rho t^* \quad \text{if } u = \rho t \lambda \text{ is not well-formed.}$$

Theorem 3. *Φ_n is $n+1$ to 1 onto B_{n0} and is bijective on each B_{ni}.*

Corollary 4. (Chung and Feller) $\text{card}(B_{ni}) = \text{card}(B_{n0})$.

Corollary 5. *If w is a random variable distributed uniformly in B_n, then $\Phi_n(w)$ is distributed uniformly in B_{n0}.*

Proof. It is sufficient to show that Φ_n is a bijection $B_{ni} \to B_{n0}$ for each i. Suppose that $w_1, w_2 \in B_{ni}$ and that they have the same image under Φ_n. For $k = 1,2$ put $w_k = u_k v_k$, where u_k irreducible, and let $|u_k| = r_k$, $|v_k| = s_k$. There are four possibilities:

(1) u_1, u_2 are each well-formed. Then $v_1 \in B_{s_1,i}$, $v_2 \in B_{s_2,i}$, and $u_1 \Phi_{s_1}(v_1) = \Phi_n(w_1) = \Phi_n(w_2) = u_2 \Phi_{s_2}(v_2)$. By Lemma 2, $u_1 = u_2$, $\Phi_{s_1}(v_1) = \Phi_{s_2}(v_2)$ and so $v_1 = v_2$ by induction.

(2) Neither of u_1 and u_2 are well-formed, say $u_1 = \rho t_1 \lambda$ and $u_2 = \rho t_2 \lambda$. Then $v_1 \in B_{s_1,i-r_1}$, $v_2 \in B_{s_2,i-r_2}$, and $\lambda \Phi_{s_1}(v_1) \rho t_1^* = \lambda \Phi_{s_2}(v_2) \rho t_2^*$. The leading subwords $\lambda \Phi_{s_1}(v_1) \rho t_1^*$ and $\lambda \Phi_{s_2}(v_2) \rho t_2^*$ are irreducible, therefore equal. Therefore, by induction, $v_1 = v_2$ and $t_1 = t_2$, so $u_1 = u_2$.

(3) u_1 is well-formed and u_2 is not well-formed, say $u_2 = \rho t_2 \lambda$. Then $v_1 \in B_{s_1,i}$, $v_2 \in B_{s_2,i-r_2}$, and $u_1 \Phi_{s_1}(v_1) = \lambda \Phi_{s_2}(v_2) \rho t_2^*$. By Lemma 2 again, $u_1 = \lambda \Phi_{s_2}(v_2) \rho$ and, taking lengths, $r_1 = s_2 + 2$ and $s_1 = r_2 - 2$. But $r_2 \leqslant i$ since $v_2 \in B_{s_2,i-r_2}$ and $i \leqslant s_1$ since $v_1 \in B_{s_1,i}$ from which it follows that $i \leqslant s_1 \leqslant r_2 - 2 < r_2 \leqslant i$, a contradiction.

(4) u_1 is not well-formed and u_2 is well-formed. This case is impossible for the same reasons as case 3.

This proves that Φ_n is one-to-one on B_{ni}. To prove that it maps B_{ni} onto B_{n0} it is enough to show that these sets have the same size. But we have seen that card $(B_{ni}) \leqslant \text{card}(B_{n0})$ for each i and if any of these inequalities were strict we would have the contradiction

$$\binom{2n}{n} = \text{card}(B_n) = \sum_{i=0}^{n} \text{card}(B_{ni}) < (n+1)\text{card}(B_{n0}) = \binom{2n}{n}. \qquad \square$$

Lemma 6. *If $w \in B_n$, $\Phi_n(w)$ can be determined in $O(n)$ time.*

Proof. We follow the inductive definition of $\Phi_n(w)$. Let $T(n)$ be the total number of operations required. The decomposition $w = uv$, where u is irreducible, can be found in $O(r)$ steps, where $r = |u|$ by partial summation; the first partial sum equal to zero defines u. Then $\Phi_{n-r}(v)$ must be calculated and so we obtain the recurrence $T(n) = O(r) + T(n-r) = O(n)$. $\square$

Note that the computation of $\Phi_n(w)$ requires no integers larger than $2n$. Thus step 3 of our algorithm can be implemented in linear time, with integers of size $O(n)$, by applying the function Φ_n.

The above discussion provides the proof of the following theorem:

Theorem 7. *Algorithm RANDOM BRACKET SEQUENCE is an unbiased random binary tree generator. It is executes in linear time and uses integers of size $O(n)$.*

Acknowledgment

We acknowledge helpful discussions with D. Horrocks, A. Knight and U. Martin.

References

[1] D.B. Arnold and M.R. Sleep, Uniform random number generation of n balanced parenthesis strings, *ACM Trans. Programming Languages Systems* **2** (1980) 122–128.

[2] K.L. Chung and W. Feller, Fluctuations in coin-tossing, *Proc. Nat. Acad. Sci. U.S.A.* **35** (1949) 605–608.

[3] W. Feller, *An Introduction to Probability Theory and its Applications*, Vol. 1 (Wiley, New York, 3rd ed., 1968).

[4] G.D. Knott, A numbering system for binary trees, *Comm. ACM* **20** (1977) 113–115.

[5] D.E. Knuth, *Semi-numerical Algorithms, The Art of Computer Programming*, Vol. 2 (Addision-Wesley, Reading, MA, 2nd ed., 1981).

[6] H.W. Martin and B.J. Orr, A random binary tree generator, in: *Computing Trends in the 1990's, ACM Seventeenth Computer Science Conf.*, Louisville, KY (ACM, New York, 1989) 33–38.

[7] E.M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory and Practice* (Prentice Hall, Englewood Cliffs, NJ, 1977).

[8] M. Solomon and R.A. Finkel, A note on enumerating binary trees, *J. ACM* **27** (1980) 3–5.