GENERATING BINARY TREES AT RANDOM

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

SCS-TR-184, DECEMBER 1990

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

GENERATING BINARY TREES AT RANDOM

M.D. ATKINSON and J.-R. SACK

School of Computer Science, Carleton University, Ottawa, CANADA K1S 5B6

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; the algorithm uses numbers no bigger than n.

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 [3, 12] and [5] 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 first overcome by Martin and Orr [5] using inversion tables; they gave a linear time algorithm which used numbers no bigger than 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: 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 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 Figure 1).

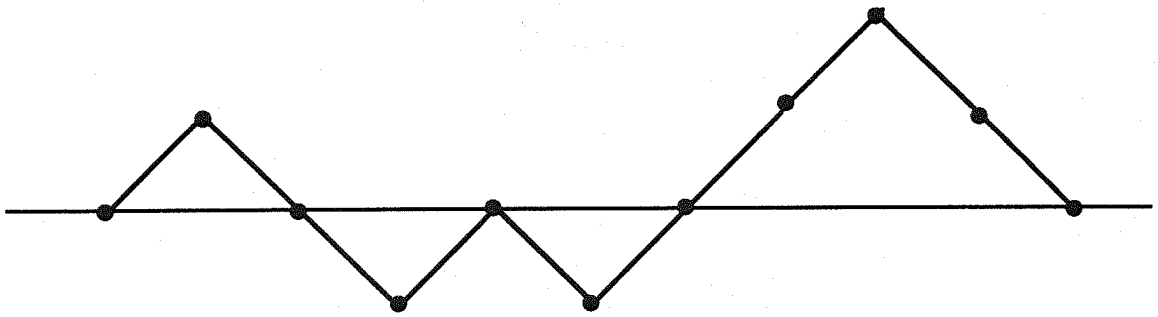

Figure 1: The zigzag diagram corresponding to the word $\lambda\rho\rho\lambda\rho\lambda\lambda\lambda\rho\rho$.

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 $w=w_1w_2$ with w_1, w_2 each balanced and non-empty, 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 sometimes called *well-formed* and correspond to well-nested bracket sequences. Observe that the defect of a word is easily found by a summation: we scan the word from left to right regarding each λ as +1, each ρ as -1, and computing the partial sums; the final sum is zero and the number of negative interim sums is 2i-1, where i 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 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 w is well-formed then $\lambda u\rho$ is irreducible.

LEMMA 2 A balanced word w has a unique factorisation as $w=w_1w_2...w_k$ 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}, ..., B_{nn}. The Chung-Feller Theorem, see [1, Theorem 2A] and [2, 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,....,2n}

2. Define a random member $x=(x_1x_2....x_{2n})$ 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,...,2n}uniformly at random, and hence a member of B_n, we may use the technique given in [4, p.137]. This technique takes only linear time and uses only numbers 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 $\Phi_n: B_n \rightarrow B_{n0}$. 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, $|u|=r>0$, $|v|=s\geq 0$; then we define Φ_n by the rules

$\Phi_n(w) = u\Phi_s(v)$ if u is well-formed

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

THEOREM 1 Φ_n is n+1 to 1 onto B_{n0} and is bijective on each B_{ni}.

COROLLARY 1 (Chung-Feller) $card(B_{ni}) = card(B_{n0})$.

COROLLARY 2 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 $B_{ni} \rightarrow B_{n0}$ 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 $w_k = u_kv_k$, where u_k irreducible, and let $|u_k| = r_k$, $|v_k| = s_k$. There are 4 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 $r_2 \le i$ since $v_2 \in B_{s_2,i-r_2}$ and $i \le s_1$ since $v_1 \in B_{s_1,i-r_1}$ from which it follows that $r_2 \le s_1$, 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 $\text{card}(B_{ni}) \le \text{card}(B_{n0})$ for each i and if any of these inequalities were strict we would have the contradiction

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

LEMMA 3 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 $T(n) = O(r) + T(n-r) = O(n)$.

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

The above discussion provides the proof of the following

THEOREM 2 Algorithm RANDOM BRACKET SEQUENCE is an unbiased random binary tree generator. It executes in linear time and uses integers no larger than n.

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

References

[1] Chung, K.L. and Feller, W.: Fluctuations in coin-tossing, Proc. Nat. Acad. Sci. USA 35 (1949), 605-608.

[2] Feller, W.: An Introduction to Probability Theory and its Applications, vol. 1 (3rd edition), John Wiley, New York - London - Sydney, 1968.

[3] Knott, G.D.: A numbering system for binary trees, Communications of the ACM 20 (1977), 113-115.

[4] Knuth, D.E.: Semi-numerical Algorithms, The Art of Computer Programming, vol 2, (2nd edition), Addison-Wesley, Reading, Massachusetts, 1981.

[5] Martin, H.W. and Orr, B.J.: A random binary tree generator, Computing Trends in the 1990's, ACM Seventeenth Computer Science Conference, Louisville, Kentucky (February 1989), ACM Press, 33-38.

[6] Solomon, M. and Finkel, R.A.: A note on enumerating binary trees, Journal of the ACM 27 (1980), 3-5.

School of Computer Science, Carleton University
Bibliography of Technical Reports

SCS-TR-141 **Graphically Defining Simulation Models of Concurrent Systems**
H. Glenn Brauen and John Neilson, September 1988

SCS-TR-142 **An Algorithm for Distributed Mutual Exclusion on Arbitrary Networks**
H. Glenn Brauen and John E. Neilson, September 1988

SCS-TR-143 to 146 are unavailable.

SCS-TR-147 **On Transparently Modifying Users' Query Distributions**
B.J. Oommen and D.T.H. Ng, November 1988

SCS-TR-148 **An O(N Log N) Algorithm for Computing a Link Center in a Simple Polygon**
H.N. Djidjev, A. Lingas and J.-R. Sack, July 1988
Available in STACS 89, 6th Annual Symposium on Theoretical Aspects of Computer Science, Paderborn, FRG, February 16-18, 1989, Lecture Notes in Computer Science, Springer-Verlag No. 349

SCS-TR-149 **Smallscript: A User Programmable Framework Based on Smalltalk and Postscript**
Kevin Haaland and Dave Thomas, November 1988

SCS-TR-150 **A General Design Methodology for Dictionary Machines**
Frank Dehne and Nicola Santoro, February 1989

SCS-TR-151 **On Doubly Linked List ReOrganizing Heuristics**
D.T.H. Ng and B. John Oommen, February 1989

SCS-TR-152 **Implementing Data Structures on a Hypercube Multiprocessor, and Applications in Parallel Computational Geometry**
Frank Dehne and Andrew Rau-Chaplin, March 1989

SCS-TR-153 **The Use of Chi-Squared Statistics in Determining Dependence Trees**
R.S. Valiveti and B.J. Oommen, March 1989

SCS-TR-154 **Ideal List Organization for Stationary Environments**
B. John Oommen and David T.H. Ng, March 1989

SCS-TR-155 **Hot-Spot Contention in Binary Hypercube Networks**
Sivarama P. Dandamudi and Derek L. Eager, April 89

SCS-TR-156 **Some Issues in Hierarchical Interconnection Network Design**
Sivarama P. Dandamudi and Derek L. Eager, April 1989

SCS-TR-157 **Discretized Pursuit Linear Reward-Inaction Automata**
B.J. Oommen and Joseph K. Lanctot, April 1989

SCS-TR-158 (revised) **Parallel Fractional Cascading on a Hypercube Multiprocessor**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, May 1989 (Revised April 1990)

SCS-TR-159 **Epsilon-Optimal Stubborn Learning Mechanisms**
J.P.R. Christensen and B.J. Oommen, June 1989

SCS-TR-160 **Disassembling Two-Dimensional Composite Parts Via Translations**
Doron Nussbaum and Jörg-R. Sack, June 1989

SCS-TR-161 (revised) **Recognizing Sources of Random Strings**
R.S. Valiveti and B.J. Oommen, January 1990
Revised version of SCS-TR-161 "On the Data Analysis of Random Permutations and its Application to Source Recognition", published June 1989

SCS-TR-162 **An Adaptive Learning Solution to the Keyboard Optimization Problem**
B.J. Oommen, R.S. Valiveti and J. Zgierski, October 1989

SCS-TR-163 **Finding a Central Link Segment of a Simple Polygon in O(N Log N) Time**
L.G. Alexandrov, H.N. Djidjev, J.-R. Sack, October 1989

SCS-TR-164 **A Survey of Algorithms for Handling Permutation Groups**
M.D. Atkinson, January 1990

SCS-TR-165 **Key Exchange Using Chebychev Polynomials**
M.D. Atkinson and Vincenzo Acciaro, January 1990

SCS-TR-166 **Efficient Concurrency Control Protocols for B-tree Indexes**
Ekow J. Otoo, January 1990

SCS-TR-167 **A Hierarchical Stochastic Automaton Solution to the Object Partitioning Problem**
B.J. Oommen, January 1990

SCS-TR-168 **Adaptive List Organizing for Non-stationary Query Distributions. Part I: The Move-to-Front Rule**
R.S. Valiveti and B.J. Oommen, January 1990

SCS-TR-169 **Trade-Offs in Non-Reversing Diameter**
Hans L. Bodlaender, Gerard Tel and Nicola Santoro, February 1990

SCS-TR-170 **A Massively Parallel Knowledge-Base Server using a Hypercube Multiprocessor**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990

SCS-TR-171 **Parallel Processing of Quad Trees on the Hypercube (and PRAM)**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990

SCS-TR-172 **A Note on the Load Balancing Problem for Coarse Grained Hypercube Dictionary Machines**
Frank Dehne and Michel Gastaldo, May 1990

SCS-TR-173 **Self-Organizing Doubly-Linked Lists**
R.S. Valiveti and B.J. Oommen, May 1990

SCS-TR-174 **A Presortedness Metric for Ensembles of Data Sequences**
R.S. Valiveti and B.J. Oommen, May 1990

SCS-TR-175 **Separation of Graphs of Bounded Genus**
Ljudmil G. Aleksandrov and Hristo N. Djidjev, May 1990

SCS-TR-176 **Edge Separators of Planar and Outerplanar Graphs with Applications**
Krzystof Diks, Hristo N. Djidjev, Ondrej Sykora and Imrich Vrto, May 1990

SCS-TR-177 **Representing Partial Orders by Polygons and Circles in the Plane**
Jeffrey B. Sidney and Stuart J. Sidney, July 1990

SCS-TR-178 **Determining Stochastic Dependence for Normally Distributed Vectors Using the Chi-squared Metric**
R.S. Valivetei and B.J. Oommen, July 1990

SCS-TR-179 **Parallel Algorithms for Determining K-width- Connectivity in Binary Images**
Frank Dehne and Susanne E. Hambrusch, September 1990

SCS-TR-180 **A Workbench for Computational Geometry (WOCG)**
P. Epstein, A. Knight, J. May, T. Nguyen, and J.-R. Sack, September 1990

SCS-TR-181 **Adaptive Linear List Reorganization under a Generalized Query System**
R.S. Valiveti, B.J. Oommen and J.R. Zgierski, October 1990

SCS-TR-182 **Breaking Substitution Cyphers using Stochastic Automata**
B.J. Oommen and J.R. Zgierski, October 1990

SCS-TR-183 **A New Algorithm for Testing the Regularity of a Permutation Group**
V. Acciaro and M.D. Atkinson, November 1990

SCS-TR-184 **Generating Binary Trees at Random**
M.D. Atkinson and J.-R. Sack, December 1990