A New Algorithm for Generating Binary Trees using Rotations

M. C. ER
Department of Computer Science, The University of Western Australia, Nedlands, WA 6009, Australia

A new formulation for generating all binary trees with n nodes by using tree rotations is presented. Such a new formulation is based on a detailed study of the properties of codewords with (n − 1) non-negative integers that are called rotational admissible. A 1 − 1 correspondence between a set of binary trees with n nodes and a set of rotational admissible codewords with (n − 1) integers is formally established. As a result, simple and efficient algorithms for generating rotational-admissible codewords and for generating binary trees using rotations are derived. An empirical test reveals that both algorithms run faster than the corresponding Zerling's algorithms for performing the same task.

Received October 1986, revised May 1987

1. INTRODUCTION

The combinatorial problem of generating binary trees is concerned with generating rooted ordered binary trees with n nodes in some orders. Many ingenious generating algorithms[1, 3, 5–9] for performing this task have been discovered. In a recent paper,[10] Zerling reported a new approach for generating binary trees by using rotations. His basic idea is to perform all possible right rotations of subtrees at all nodes, starting from a left linear tree. By encoding the number of right rotations at each node in a codeword, the relation between a binary tree and a codeword can be established. However, Zerling's claim that the set of binary trees with n nodes and the set of codewords are in $1-1$ correspondence is imprecise because the set of codewords has not been formally defined. In other words, we are not certain whether or not there exists a codeword such that there is no corresponding binary tree.

In what follows, we take a new look at the set of codewords, and provide a new formulation for generating the set of binary trees with n nodes by using tree rotations.

2. DEFINITIONS AND NOTATIONS

Let $T(n)$ be a set of binary trees with n nodes. Assume $T \in T(n)$. We say that T is a left linear tree if all its n nodes form a linear list such that each node is the left child of its parent, starting from the root. By the left arm L_T of T, we mean the path from the root to the leftmost leaf of T. Similarly, the right arm R_T of T is the path from the root to the rightmost leaf of T. Further, by node i of T, we mean the ith node of the left arm of T (here, we assume that node 1 of T is the root of T).

Following the conventions used in tree balancing, we use the terms right rotation and left rotation rather than unfolding and folding as used by Zerling.[9] Let $|L_T|$ and $|R_T|$ denote the numbers of nodes on the left arm and the right arm of T, respectively. The effect of performing a right rotation on a tree T is to decrement $|L_T|$ by one and to increment $|R_T|$ by one. Conversely, the left rotation is precisely the reverse of the right rotation. An example may be seen in Fig. 1.

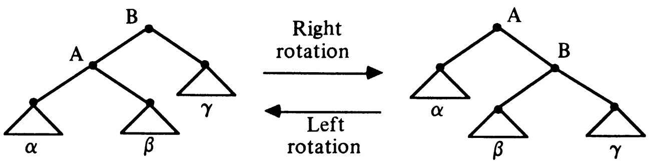

Figure 1. An example of left and right rotations.

Definition 2.1. (Rotational admissible)

Let $X = x_{n-1}x_{n-2}\ldots x_2x_1$ be a codeword of $(n-1)$ non-negative integers. A codeword is called rotational admissible if it satisfies:

(i) $0 \leqslant m \leqslant n-1$, where $m = \sum_{i=1}^{n-1} x_i$; and

(ii) $0 \leqslant x_i \leqslant n-i-\sum_{j=i+1}^{n-1} x_j$. $\square$

Let $X(n-1)$ be a set of rotational-admissible codewords, each of which consists of $(n-1)$ non-negative integers. The sets $X(n-1)$ and $T(n)$ will be shown to be in $1-1$ correspondence in the next section.

3. MAPPING BETWEEN *X*(*n* − *1*) AND *T*(*n*)

Let $|T(n)|$ be the number of distinct binary trees with n nodes. It is well known[3] that:

$$|T(n)| = \frac{1}{n+1}\binom{2n}{n}$$

which is also called the nth Catalan number.

Theorem 3.1.

$$|X(n-1)| = \frac{1}{n+1}\binom{2n}{n}.$$

Proof.

Let $X = x_{n-1}x_{n-2}\ldots x_2x_1$ be a member of $X(n-1)$, and let

$$m = \sum_{i=1}^{n-1} x_i$$

First of all, we relate x to a lattice path. Suppose X signifies a staircase path from $(1,0)$ to (n,m) in the PQ-plane, such that x_i indicates the relative height of the $(n-i)$th stair with respect to the previous one, and the relative height of the first stair is defined with respect to the stair connecting $(0,0)$ and $(1,0)$. For example, the

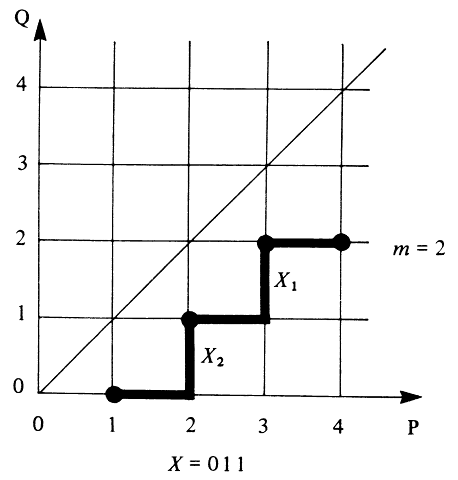

Figure 2. Codeword $X = 011$ and its related staircase path in the PQ-plane, when $n = 4$.

codeword $X = 011$ and its related staircase path are shown in Fig. 2, assuming that $n = 4$. Since X is rotational admissible, by definition, the staircase path of X is bounded by the positive P-axis and the line $p = q$. Therefore, by the Reflection Principle,[2] the total number of legal paths connecting $(1,0)$ and (n, m) bounded by lines $q = 0$ and $p = q$ is given by

$$\binom{n-1+m}{m} - \binom{n-1+m}{m-1}.$$

But $m = 0, 1, \ldots, n-1$, hence

$$|X(n-1)| = \sum_{m=0}^{n-1}\left(\binom{n-1+m}{m} - \binom{n-1+m}{m-1}\right).$$

After simplification, the theorem is proven. □

Theorem 3.2.

The mapping between $X(n-1)$ and $T(n)$ is $1-1$.

Proof.

Starting from a left linear tree $T \in T(n)$ of n nodes, x_i of $X \in X(n-1)$ may be interpreted as the number of right rotations applied to the node i of T. The effect of a right rotation is to reduce the length of the left arm L_T of T by one. For a left linear tree T, initially $|L_T| = n$; thus the maximum number of right rotations that can be performed on T is $n-1$. Hence $m \leqslant n-1$, where

$$m = \sum_{i=1}^{n-1} x_i.$$

But the minimum number of right notations is 0. Therefore, $0 \leqslant m \leqslant n-1$. This is the first condition stated in Definition 2.1.

Further, the maximum number of right rotations allowed at the node i of the left linear tree is $(n-i)$. If right rotations $x_{n-1}, x_{n-2}, \ldots, x_{i+1}$ are applied to the subtree at node i, the maximum number of right rotations at node i is reduced by

$$\sum_{j=i+1}^{n-1} x_j.$$

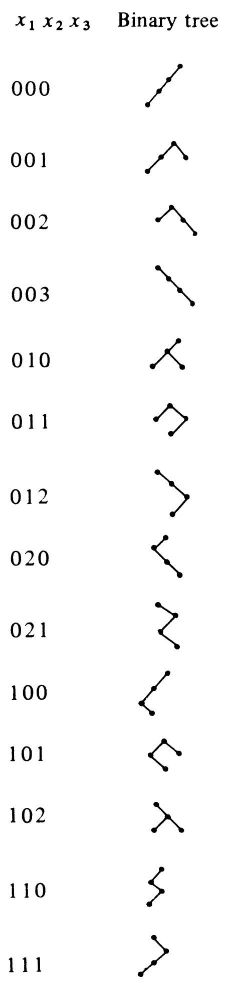

Figure 3. The $1-1$ correspondence between $X(3)$ and $T(4)$.

Therefore, the second condition in Definition 2.1,

$$0 \leqslant x_i \leqslant (n-i) - \sum_{j=i+1}^{n-1} x_j$$

is satisfied.

Thus, to find the corresponding tree T of a codeword $X = x_{n-1}x_{n-2}\ldots x_2x_1$, node i of a n-node left linear tree is rotated rightward x_i times, for i from $(n-1)$ to 1. Conversely, to find the codeword of a given tree, the tree is rotated leftward until it becomes a left linear tree, from node 1 to node $(n-1)$. The maximum number of left rotations performed at node i is x_i. □

An example of the $1-1$ correspondence of $X(3)$ and $T(4)$ is shown in Fig. 3.

4. GENERATING ALGORITHMS

Equipped with Theorem 3.2, an efficient algorithm for generating all binary trees with n nodes can be constructed. First of all, we describe an efficient algorithm for generating all rotational-admissible codewords of $X(n-1)$. The basic idea is based on Definition 2.1. We see that

$$\sum_{i=1}^{n-1} x_i = m$$

such that $0 \leqslant m \leqslant n-1$. So the topmost control of the algorithm is to generate all possible values of m from 0 to $n-1$ inclusively. Further, we see that whenever x_i (for *i* from $n-1$ to 1) is generated, the value of *m* is reduced by x_i so that by the time x_1 is generated, *m* is reduced to zero. Thus, from the above information, we derive the algorithm for generating $X(n-1)$ in the lexicographic order.

```
procedure Generate(i, m: integer);
var j: integer;
begin
  if i < = 0 then OutputCodeword
  else for j: = 0 to m - i + 1 do begin
    x[i]: = j;
    Generate(i - 1, m - j);
    end;
end {Generate};
```

Here we assume that $x[1\,..\,n-1]$ is declared as a global array as follows:

var $x[1\,..\,n-1]$: *codeword*;

Note that the procedure call, *OutputCodeword*, is simply to print the contents of the codeword $x[1\,..\,n-1]$ in the reverse order. The above generating algorithm should be called as *Generate*$(n-1, n-1)$.

Once the above generating algorithm is available, it can be modified to generate binary trees. The simplest approach is to modify the procedure *OutputCodeword* so that it converts a generated codeword to a binary tree. Starting from a left linear tree of *n* nodes, x_i right rotations are applied to node *i* of this tree, for *i* from $n-1$ to 1. The resulting tree is the corresponding tree of a generated codeword.

A better approach is to integrate tree rotations with the generation of codewords. When a non-zero value is assigned to x_i, node *i* of a tree (initially starting from a left linear tree) is rotated rightward accordingly. As soon as the maximum value of x_i is reached, the right subtree at node *i* is rotated leftward x_i times to reset the subtree at node *i* to its initial starting shape. By induction on *i* from $n-1$ to 1, an algorithm for generating all shapes of binary trees with *n* nodes is apparent. Assuming the following global declaration:

var $x[1\,..\,n-1]$: *codeword*;

the detailed algorithm is presented below.

```
procedure GenTrees(i, m: integer);
var j: integer;
begin
  if i < = 0 then {a tree has been formed}
  else begin
    x[i]: = 0;
    GenTrees(i - 1, m);
    for j: = 1 to m - i + 1 do begin
      x[i]: = j;
      perform one right notation at node i;
      GenTrees(i - 1, m - j);
      end;
    perform (m - i + 1) left rotations at node i;
    end;
end {GenTrees};
```

Again, the above procedure is activated as *GenTrees*$(n-1, n-1)$.

For each right rotation of a tree, there is a left rotation to reset it in the course of generating all binary trees. Hence the procedure *GenTrees* simply doubles the work of *Generate*.

5. EFFICIENCY OF GENERATING ALGORITHMS

To compare the efficiency of our *Generate* algorithm with Zerling's *EXAMINE* algorithm,[10] both algorithms have been coded in PASCAL and tested on a VAX 11/750 computer running under UNIX 4.2 BSD. The actual running times (measured in seconds of CPU time) of both algorithms, averaged over three runs, are summarised in Table 1.

Table 1. A comparison of the average running times of the algorithms *EXAMINE* and *Generate* on a VAX 11/750 computer (measured in seconds of CPU time)

n	Zerling's EXAMINE	Generate
3	0.050	0.047
4	0.070	0.050
5	0.153	0.147
6	0.420	0.333
7	1.293	1.040
8	4.173	3.387
9	13.977	11.263
10	47.897	38.150

The result in Table 1 reveals that our *Generate* algorithm consistently runs faster than Zerling's *EXAMINE* algorithm.[10] Since both algorithms are adapted for generating binary trees, our algorithm *GenTrees* thus generates all binary trees with *n* nodes faster than Zerling's modified *EXAMINE* algorithm for performing the same task. Further, Zerling's algorithm is found to be erroneous when $n = 1$.

Moreover, a comparison of our generating algorithm with Zerling's *EXAMINE* algorithm[10] reveals that our algorithm is simpler and uses less array space.

Acknowledgement

The author thanks the referee who made helpful suggestions that improved the presentation of this paper.

REFERENCES

1. M. C. Er, A note on generating well-formed parenthesis strings lexicographically. *The Computer Journal* **26**, 205–207 (1983).
2. W. Feller, *An Introduction to Probability Theory and its Applications*, vol. 1. Wiley, New York (1968).
3. G. D. Knott, A numbering system for binary trees. *Commun. ACM* **20**, 113–115 (1977).
4. D. E. Knuth, *The Art of Computer Programming*, vol. 1, *Fundamental Algorithms*. Addison-Wesley, Reading, Mass. (1973).

5. A. Proskurowski, On the generation of binary trees. *J. ACM* **27**, 1–2 (1980).
6. D. Rotem and Y. L. Varol, Generation of binary trees from ballot sequences. *J. ACM* **25**, 396–404 (1978).
7. F. Ruskey and T. C. Hu, Generating binary trees lexicographically. *SIAM J. Computing* **6**, 745–758 (1977).
8. M. Solomon and R. A. Finkel, A note on enumerating binary trees. *J. ACM* **27**, 3–5 (1980).
9. S. Zaks, Lexicographic generation of ordered trees. *Theor. Comput. Sci.* **10**, 63–82 (1980).
10. D. Zerling, Generating binary trees using rotations. *J. ACM* **32**, 694–701 (1985).

Announcement

9–13 July 1990

WCCE/90, Fifth World Conference on Computers in Education, Sydney, Australia

Conference Programme

Start planning now for the Fifth World Conference on Computers in Education. Such conferences are only held every five years. The conference will address **all** aspects of educational computing, ranging across primary, secondary, tertiary, industry as well as community education. It will provide an opportunity for people from many countries and backgrounds to meet and exchange ideas and experiences.

WCCE/90's six major streams

(1) Informatics Education at the Secondary Level (WG 3.1), Stream Chairman: Peter Bollerslev, Denmark
(2) Advanced Curriculum Projects in Information Processing (WG 3.2), Stream Chairman: Professor William Atchison, USA
(3) Research on Educational Applications of Information Technologies (WG 3.3), Stream Chairman: Professor Robert Lewis, UK
(4) Vocational Education and Training (WG 3.4), Stream Chairman: Dr Ben Sion Barta, Israel
(5) Informatics in Elementary Education (WG 3.5), Stream Chairman: Frank Lovis, UK
(6) Distance Learning (WG 3.6), Stream Chairman: Gyozo Kovacs, Hungary

Other activities at WCCE/90

Panel and Poster Sessions, Workshops and Meetings

Opportunities will be provided during WCCE/90 for the presentation and discussion of ideas and experiences in informal panel discussions and poster sessions. The weekend prior to the conference will be the time for some practical computer education workshops, including some run by keynote and invited speakers. During the conference there will be scheduled public meetings to form new, and rejuvenate old, interest groups.

The Galah Sessions

The Galah, a member of the Cockatoo family, is an Australian bird well known for its habit of gathering in large, noisy flocks. Our early morning and late afternoon 'Galah sessions' will provide you with the opportunity of meeting people from around the world who share your special interests in computer education.

The Exhibition

An international commercial exhibition will also be part of WCCE/90.

Mini-conferences during WCCE/90

In addition to six major streams, WCCE/90 will provide a venue for mini-conferences, including:

Teleteaching 90

Programme Chairman: Gordon Davies, UK

Teleteaching 90 is the second in a series of international conferences addressing the topic of educational technology for distance learning. Teleteaching 86 in Budapest was the first in the series. Teleteaching 90 will include papers, invited speakers, discussion panels and demonstrations.

CBT/90

Programme Chairman: Rod Sims, Australia

Hosted by the Australian Society for Computers in Learning in Tertiary Education, CBT/90 will provide an opportunity for training and education professionals to share their experiences in the design, development and delivery of computer-based instructional resources. CBT/90 will include presentations on the latest courseware applications in business, industry and government, as well as on the latest authoring tools and presentation technologies.

Kids and Classrooms 90

Programme Chairman: Pam Gibbons, Australia

An exciting programme involving children and computers is being planned by the New South Wales Computer Education Group. Included as part of WCCE/90, it will provide an opportunity for delegates to visit 'Fly on the Wall Classrooms' at the conference, where local teachers will conduct lessons with local students. For student delegates to the conference there will be 'All hands-on' sessions, where they will have the opportunity to work with computers alongside leading local teachers. This is certain to be an attraction for those overseas delegates incorporating the conference into a family vacation. Both overseas and local students are invited to contribute to the 'Kids and Classrooms Corner', where (K–12) students will present their work with computers in their schools.

'Be our guest'

WCCE/90 will offer two special schemes for overseas delegates. For those on tight budgets, there will be 'Australian Home Hospitality', providing free accommodation in Australian homes during the conference. The Education Visits Scheme will enable overseas delegates to visit schools, colleges and universities in Sydney and elsewhere before and after the conference. Because our main school holidays are in January, not July, many schools will be open at the time of your visit to Australia.

The Pacific focus of WCCE/90: New Zealand and Japan

A Pre-Conference in New Zealand and a Post-Conference in Japan will provide international delegates with the chance to obtain maximum value from international airfares, with visits to other Pacific Rim destinations.

Computers in Education: National Perspectives (New Zealand)

Programme Chairman: Ian Mitchell, New Zealand

The New Zealand event, hosted by the New Zealand Computer Society, will be held on 4–6 July. Selected papers will form the basis of workshops, the objective of which will be the production of a 'Working Document' of conclusions and recommendations on national policies for computers in education. Particular emphasis will be given to educational philosophies and structures that harmonise high technology with the culture of minority indigenous peoples.

The workshops will be held within the traditional procedures and protocols of the Maori people with their emphasis on consensus and harmony and of living and eating together during the conference.

Advanced Research on Computers in Education (Japan)

Programme Chariman: Professor Setsuko Otsuki, Japan

Hosted by the Information Processing Society of Japan and to be held on 18–20 July, the aim of this post-WCCE/90 mini conference is to explore the application of the latest developments in information processing to the field of education.

For Further Information

If you are interested in finding out more about WCCE/90, please write to us. You will be placed on our mailing list for the free WCCE/90 Newsletter and will be sent the conference registration forms in late 1989.
All communications regarding WCCE/90 should be addressed to:
WCCE/90, PO Box 319, Darlinghurst, NSW 2010, Australia. Tel: (+612) 211-5855. Fax (+612) 281-1208.