Uniform Generation of Rooted Ordered Trees with Prescribed Degrees

M. D. ATKINSON
School of Computer Science, Carleton University, Ottawa K1S 5B6, Canada

An efficient algorithm is given for generating uniformly at random a rooted tree with a specified number of nodes of each degree. The algorithm requires linear time, needs hardly any auxiliary storage and uses only very simple operations.

We shall consider rooted, ordered trees on n nodes and how to generate them uniformly at random. One natural approach is to define a listing of the set of trees, generate a random index into this list and construct the tree which has this index. The problem with this approach is that numbers which are exponential in n are required. This difficulty was pointed out by Martin and Orr [2] in the case of binary trees. They gave an algorithm for generating a binary tree uniformly at random which used only $O(n)$ arithmetic operations on integers of magnitude $O(n)$. An earlier and simpler algorithm with these features was given by Arnold and Sleep [1]. The purpose of this note is to point out how a theorem of Raney [4] can be used to solve a much more general problem equally efficiently: how to generate uniformly at random a tree where the number of nodes of each down degree is specified in advance.

We define the *type* of a tree on n nodes to be the vector $(d_0, d_1, \ldots, d_{n-1})$ where d_r is the number of nodes with r children. Of course, $\Sigma\, d_r = n$ but the type vector must also satisfy a consistency condition

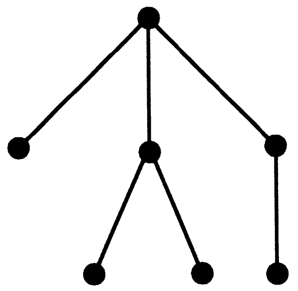
$$\sum_{r=0}^{n-1} (r-1)d_r = -1 \tag{1}$$

which expresses that the number of vertices in the tree exceeds the number of edges by 1.

We shall give an algorithm, of linear time complexity in the number of nodes n, that uses only integers of magnitude $O(n)$ and which, given a type vector, generates a tree of that type uniformly at random. The special case $d_0 = N+1$, $d_2 = N$, $d_r = 0$ otherwise, corresponds to binary trees with N internal nodes all of down-degree 2; we call these *full* binary trees. The operation of deleting the leaves of a full binary tree on N internal nodes defines a one-to-one correspondence with binary trees on N nodes so our algorithm solves the case considered in [1, 2]. In a similar way, putting $d_0 = (k-1)N+1$, $d_k = N$, and $d_r = 0$ otherwise, we can generate k-ary trees uniformly at random.

We shall represent a tree by the pre-order listing of the down degrees of its nodes. Clearly this sequence of integers uniquely defines the tree. If a sequence of integers can arise in this way we shall say that it is *valid*. A valid sequence must either consist of zero alone or must consist of a positive integer r followed by r valid sequences. Such sequences were considered by Raney [4] who regarded them as prefix expressions where each integer r stands for an r-ary operator; his counting technique is the basis of our algorithm.

An example valid sequence is 3 0 2 0 0 1 0 and the corresponding tree is



Clearly, if d_r is the number of occurrences of r in a valid sequence $\sigma_1\sigma_2 \ldots \sigma_n$, then equation (1) must be satisfied. This equation may be recast as $\Sigma_{i=1}^{n}(\sigma_i - 1) = -1$. It is not a sufficient condition on a string of integers from $\{0, 1, \ldots, n-1\}$ to be a valid sequence. One needs also the following inequalities which are easily proved by induction.

$$\sum_{i=1}^{m} (\sigma_i - 1) \geqslant 0 \quad \text{for all} \quad 0 \leqslant m < n \tag{2}$$

Raney [4, Theorem 2.1] proved the following striking result:

THEOREM 1. If $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$ is any string of symbols from $\{0, 1, \ldots, n-1\}$, d_i is the number of symbols equal to i and equation (1) holds then there is precisely one cyclic shift of σ which is a valid sequence.

It is not difficult to identify the cyclic shift of σ which produces a valid sequence. Put $\sigma'_i = \sigma_i - 1$, let $\sigma' = \sigma'_1\sigma'_2 \ldots \sigma'_n$, and let s_m denote the partial sum $\Sigma_1^m \sigma'_i$. Conditions (1) and (2) may be rewritten as

$$s_n = -1 \tag{1'}$$

$$s_m \geqslant 0 \quad \text{for} \quad 0 \leqslant m < n \tag{2'}$$

We must therefore determine the cyclic shift of σ which makes these equations valid. Let k be that index for which s_k is minimal and, if there is more than one such index, let k be the minimal one. Consider the shift of σ' which moves position $k+1$ into the first position. The partial sums of this shifted sequence are either of the form

$$\sum_{i=k+1}^{m} \sigma'_i, \qquad k+1 \leqslant m \leqslant n$$

which are all non-negative (otherwise s_k would not be the minimal partial sum) or they have the form

$$\sum_{i=k+1}^{n} \sigma'_i + s_j, \qquad j \leqslant k$$

If any of the latter expressions satisfied $\Sigma_{i=k+1}^{n} \sigma'_i + s_j \leqslant -1$ then, because $\Sigma_{i=k+1}^{n} \sigma'_i + s_k = -1$, we would have $s_j \leqslant s_k$ but, again by definition of k, the only value of j for which this can happen is $j = k$. We therefore deduce that the shift of σ which moves position $k+1$ into the first position produces a valid sequence. We call position $k+1$ the *root position* of σ. Clearly, the following computation finds the root position in time $O(n)$:

ROOT POSITION
Input: a sequence $\sigma_1 \sigma_2 \ldots \sigma_n$ satisfying the hypotheses of Theorem 1
Output: the root position of the sequence

```
minimum := 0
s := 0
for i := 1 to n do
  s := s + $\sigma_i$ − 1
  if s < minimum then
    minimum := s
    k := i
  endif
endfor
return (k mod n + 1)
```

Now consider the following algorithm:

RANDOM TREE
Input: a sequence of non-negative integers $(d_0, d_1, \ldots, d_{n-1})$ satisfying $\sum d_r = n$ and $\Sigma_{r=0}^{n-1} (r-1)d_r = -1$
Output: a valid sequence with d_r occurrences of r for each $r = 0, 1, \ldots, n-1$

```
1. Generate an integer sequence $\sigma$ of length $n$ uniformly at random which has $d_r$ occurrences of $r$ for each $r = 0, 1, \ldots, n-1$
2. Compute the root position of $\sigma$
3. Apply to $\sigma$ the rotation which brings the root position of $\sigma$ to the front
4. Return($\sigma$)
```

As a direct consequence of Theorem 1 we have

THEOREM 2. RANDOM TREE is an unbiased generator for trees of the given type $(d_0, d_1, \ldots, d_{n-1})$.

THEOREM 3. RANDOM TREE can be implemented to run in time $O(n)$ with arithmetic on integers of magnitude $O(n)$, with $O(1)$ auxiliary space. Except for the arithmetic involved in a random integer generator the only arithmetic operations are addition and subtraction.

Proof. To implement step 1 we begin by forming the sequence whose first d_0 elements are equal to 0, the next d_1 are equal to 1, and so on. We then apply a uniformly chosen random permutation to this sequence. This can be done by the well known swapping algorithm which swaps the ith member with a randomly chosen jth member ($j \leqslant i$), $i = n, n-1, \ldots, 2$.

For step 2 we use the algorithm described earlier. Finally, for step 3, we apply an algorithm for performing rotation in place such as that given in [3].

ACKNOWLEDGEMENT

I thank the referee for his comments which resulted in an improved method for computing the root position.

REFERENCES

[1] D. B. Arnold and M. R. Sleep, Uniform random number generation of n balanced parenthesis strings. *ACM Transactions on Programming Languages and Systems*, **2**, pp. 122–128 (1993).
[2] H. W. Martin and B. J. Orr, A random binary tree generator, Computing Trends in the 1990's, *ACM 17th Computer Science Conference*, Louisville, Kentucky, pp. 33–38 (1989).
[3] D. Gries, *The Science of Programming*. Springer-Verlag, New York (1981).
[4] G. N. Raney, Functional composition patterns and power series reversion. *Transactions of the American Mathematical Society*, **94**, pp. 441–451 (1960).