COMPUTER SCIENCES DEPARTMENT

University of Wisconsin - Madison

A NOTE ON ENUMERATING t-ary TREES

Samuel W. Bent

Computer Sciences Technical Report #514

October 1983

A Note on Enumerating t-ary Trees

Samuel W. Bent *
Computer Science Department
University of Wisconsin-Madison

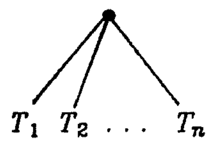
* Research partially supported by National Science Foundation Grant MCS 8203238.

Keywords: t-ary trees, enumeration, Catalan numbers

The set S_t of t-ary trees is defined for $t \geq 2$ as follows.

i. The empty tree is in S_t.

ii. If $T_1, T_2, \ldots, T_t$ are in S_t, then so is the tree



consisting of a root with t subtrees equal to $T_1, T_2, \ldots, T_n$, ordered left to right.

iii. Nothing else is in S_t.

The order among children is important, as are the gaps left by empty subtrees. Figure 1 shows the five 2-ary trees with 3 nodes; in all but one, the root has an empty subtree.

The number of t-ary trees with n nodes is well-known to be $\binom{tn}{n}\frac{1}{(t-1)n+1}$. When $t=2$ this gives the n-th Catalan number, $\binom{2n}{n}\frac{1}{n+1}$, as the number of binary trees with n nodes. These formulas have a very long history, dating back to 1758 when Euler tabulated the first few Catalan numbers [3], and to 1841 when Grünert tabulated the first few numbers for $t=3,4,5,6$ [4]. A linear recurrence for the Catalan numbers was first shown by Rodrigues [6]. Further history and a comprehensive bibliography have been published by Brown [1].

Several proofs of the "closed" formulas given above have appeared, using the Lagrange inversion formula [2,7], generating functions and a generalized binomial theorem [5, ex. 2.3.4.4-11], and direct but lengthy calculations of binomial coefficient identities [8], but the complexity of these proofs seems incommensurate with the simplicity of the result. This note presents a simple direct proof, using algorithmic and combinatorial ideas.

If $\mathbf{a}=<a_1, \ldots, a_k>$ is a sequence of integers, let $s_j(\mathbf{a})=\sum_{1\le i\le j} a_i$ be the sum of the first j elements.

The algorithm of Figure 2 produces a sequence of integers $\mathbf{a}$ from a t-ary tree by doing a preorder traversal from the root r, printing $t-1$ before visiting each leftmost subtree and printing -1 before visiting each other subtree. It also prints an extra -1 after the traversal is done. Exclusive of recursive calls, visit(v) prints t numbers that sum to 0, and summing the first $j \le t$ of these gives a nonnegative result. From this observation, the following facts about $\mathbf{a}$ follow immediately:

(a1) The sequence $\mathbf{a}$ contains $tn+1$ integers, each of which is either $t-1$ or -1.

(a2) $s_{tn+1}(\mathbf{a})=-1$.

(a3) $s_j(\mathbf{a})\ge 0$, for $1\le j\le tn$.

Conversely, from a sequence that satisfies properties (a1)-(a3), we can recover a t-ary tree with n nodes by first removing the last element (which must be -1), producing a sequence $\mathbf{b}$ satisfying

(b1) The sequence $\mathbf{b}$ contains tn integers, each either $t-1$ or -1.

(b2) $s_{tn}(\mathbf{b})=0$.

(b3) $s_j(\mathbf{b})\ge 0$, for $1\le j\le tn$.

From such a sequence, form a t-ary tree using the algorithm of Figure 3. The algorithm works by forming t subsequences of $\mathbf{b}$ starting at the places where the running sum $s_j(\mathbf{b})$ first reaches $t-1$, $t-2$,..., and 0. Since the first element of $\mathbf{b}$ must be $t-1$ and since the sum can only decrease by 1 at any step, these subsequences are well-defined. Discarding the first element of each subsequence yields t sequences satisfying (b1)-(b3), which can be recursively converted to the t subtrees of a root.

Alternatively, the algorithm amounts to parsing the sequence using the grammar

$$S \to t\text{-}1\ S\ \text{-}1\ S\ \text{-}1\ \dots\ \text{-}1\ S$$
$$S \to \varepsilon,$$

retaining the subtree of the parse tree induced by the nodes labelled S, then discarding the leaves.

Next we show that exactly one out of every $tn+1$ of the sequences satisfying (a1) and (a2) also satisfies (a3). Partition the set of sequences satisfying (a1) and (a2) into orbits under rotation; two sequences $<a_1, \dots, a_{tn+1}>$ and $<b_1, \dots, b_{tn+1}>$ are in the same orbit if and only if

$$<a_1, \dots, a_{tn+1}> = <b_j, \dots, b_{tn+1}, b_1, \dots, b_{j-1}>$$

for some j. Let $l(\mathbf{a})$ be the index where $s_j(\mathbf{a})$ first attains its minimum value. If $\mathbf{b}$ is formed by rotating the first element of $\mathbf{a}$ to the end, then

$$(1)\qquad l(\mathbf{a}) = \begin{cases} l(\mathbf{b})+1, & \text{if } 1 \le l(\mathbf{b}) \le tn, \\ 1, & \text{if } l(\mathbf{b}) = tn+1, \end{cases}$$

because

$$s_{j+1}(\mathbf{a}) = s_j(\mathbf{b}) + a_1, \quad \text{for } 1 \le j \le tn,$$
$$s_1(\mathbf{a}) = s_{tn+1}(\mathbf{b}) + a_1 + 1.$$

The partial sums of $\mathbf{b}$ (except the last) equal those of $\mathbf{a}$ offset by one and translated by a_1, so the minimum value occurs offset by one, unless it first occurs as $s_{tn+1}(\mathbf{b})$. In this case the minimum value is -1 and $s_j(\mathbf{b}) \ge 0$ for $1 \le j \le tn$, so $s_{j+1}(\mathbf{a}) = s_j(\mathbf{b}) + a_1 \ge a_1$ for $1 \le j \le tn$, and thus $l(\mathbf{a}) = 1$. Equation (1) implies that the $tn+1$ sequences in a particular orbit are all different, and that exactly one of them satisfies (a3).

A sequence satisfying (a1) and (a2) must contain exactly n elements equal to $t-1$, so there are $\binom{tn+1}{n}$ of them. By equation (1), there are $\binom{tn+1}{n}\frac{1}{tn+1}$ sequences satisfying (a1)-(a3). The algorithms of Figures 1 and 2 provide a 1-1 correspondence between t-ary trees with n nodes and sequences satisfying (a1)-(a3). Thus the number of t-ary trees with n nodes is

$$\binom{tn+1}{n}\frac{1}{tn+1}=\binom{tn+1}{(t-1)n+1}\frac{1}{tn+1}$$
$$=\binom{tn}{(t-1)n}\frac{1}{(t-1)n+1}$$
$$=\binom{tn}{n}\frac{1}{(t-1)n+1}.$$

References

1. William G. Brown, "Historical note on a recurrent combinatorial problem". *American Mathematical Monthly* **72** (1965), 973-977.
2. A. Cayley, "On the partitions of a polygon". *Proc. of the London Mathematical Society* **22** (1890-1891), 237-262.
3. L. Euler, *Novi Commentarii Acadamiae Scientarium Imperialis Petropolitanae* **7** (1758-1759), 13-14.
4. J. A. Grunert, "Über die Bestimmung der Anzahl der verschiedenen Arten, auf welche sich ein n-eck durch Diagonalen in lauter m-ecke zerlegen lässt, ...". *Archiv der Mathematic und Physik* **1** (1841), 193-203.
5. Donald E. Knuth, *The Art of Computer Programming.* Volume I: *Fundamental Algorithms*. Addison-Wesley (second edition, 1971), Reading, Mass.
6. Olinde Rodriques, "Sur le nombre de manières de décomposer un polygone en triangles au moyen de diagonales". *Journal de Mathematiques Pures et Appliquées* **3** (1838), 547-548.
7. H. M. Taylor and R. C. Rowe, "Note on a geometrical theorem". *Proc. of the London Mathematical Society* **13** (1881-1882), 102-106.
8. Anthony E. Trojanowski, "On the ordering, enumeration and ranking of k-ary trees". Report UIUCDCS-R-77-850 (1977), Dept. of Computer Science,

University of Illinois, Urbana, Illinois.

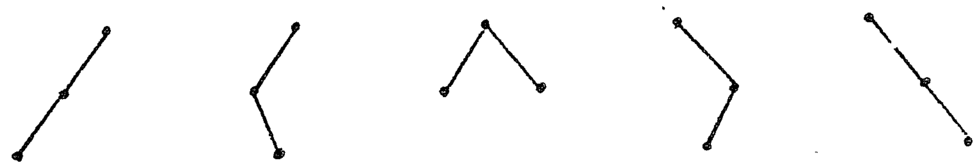


Figure 1

The 2-ary trees with 3 nodes.

```
procedure TreeToSequence(r);
 {        visit(r);
          print(-1);
          };

procedure visit(v);
 {        if v=null then return;
          print(t-1);
          visit(first child of v);
          for i:=2 to t do
            {        print(-1);
                     visit(i-th child of v);
                     };
          };
```

Figure 2

Generating a sequence from a t-ary tree.

```
procedure SequenceToTree(b);
 {      if empty(b) then return(empty-tree);
        sum:=0;  i:=0;  fence:=t-1;
        while not empty(b) do
         {      x := next(b);
                sum := sum + x;
                if sum=fence then
                 {      i:=i+1;  fence:=fence-1;
                        s_i := empty;
                        };
                else
                 {      append x to s_i;
                        };
                };
        for i:=1 to t do
         {      T_i := SequenceToTree(s_i);
                };
        return(   [tree: root with subtrees T_1 ... T_t]   );
```

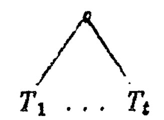


Figure 3

Generating a tree from a sequence.