Generating Edge-Labeled Trees

Oleg Pikhurko

In this note we present a bijective enumeration of all *edge-labeled trees* (where we put labels on edges, while vertices are indistinguishable from each other). These are not as important as vertex-labeled trees, but they do appear in applications (see, for example, Cameron [**3**], [**4**], Buneman et al. [**1**], Calcagno et al. [**2**], and Hage [**7**]). Cameron [**4**, Proposition 2.1] demonstrated that the total number of edge-labeled trees with n (≥ 2) edges is $(n+1)^{n-2}$ by showing that the number of vertex-labeled trees of size n is $n+1$ times larger than the number of edge-labeled ones.

However, it is often desirable not only to know the total number of objects of a given type but also to have a method for generating them in some linear ordering. For example, this is helpful when one wishes to check some hypothesis for each object. In our case, it would be ideal if we could find a bijection between the family of edge-labeled trees with n edges and X^{n-2}, where X is some standard set of size $n+1$. This would offer extra advantages, such as being able to sample uniformly distributed random edge-labeled trees by taking a random element of X^{n-2}.

There are many known bijections for vertex-labeled trees (see Prüfer [**9**], Moon [**8**], and Eğecioğlu and Remmel [**5**], to name just a few references). Combining these with Cameron's proof, one can write a bijection for edge-labeled trees, but the resulting algorithms seem rather unwieldy. Cameron [**4**, Problem 1] asked for a direct bijection (i.e., one not going through vertex-labeled trees). Here we give such a construction based on the ideas of Foata [**6**].

Fix $n \geq 2$. Let T be any tree with n edges labeled by $e_1, \dots, e_n$, where the labels are ordered $e_1 < \cdots < e_n$. Let L consist of all *leaves* (i.e., pendant edges) e_i with $1 \leq i \leq n-1$. Subdivide e_n with a new vertex x_0 into two new edges e'_n and e''_n. Here we have two choices as to how to assign these labels, so we agree that e'_n is the edge that lies on the path from x_0 to the smallest element of L.

Initially, let S be the empty sequence. Repeat the following procedure for each e_i in L, taking them in increasing order. Move along the path from e_i to x_0, naming the edges encountered $p_1 = e_i, p_2, p_3, \dots,$ until the current edge p_j is e'_n, e''_n, or an element already in S. Then stop and append the sequence $p_j, \dots, p_3, p_2$ to the end of S. Note that we add elements in the order reverse to that in which they were visited and we exclude $p_1 = e_i$.

Clearly, the result of this construction is a sequence S that starts with e'_n and has length $n - 1$. The code $C = C(T)$ of the tree T is obtained from S by deleting the first e'_n.

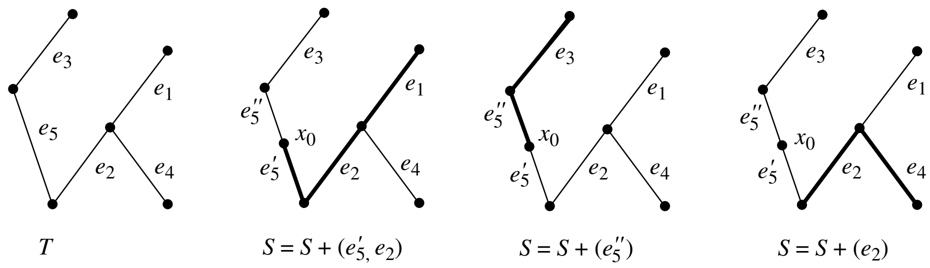

Figure 1. An illustration: computing $C(T) = (e_2, e''_5, e_2)$.

On the other hand, let C be any sequence of length $n - 2$ consisting of symbols from $\{e_1, \dots, e_{n-1}, e'_n, e''_n\}$. Obtain a new sequence S from C by adding a leading e'_n. Let L consist of those labels in $\{e_1, \dots, e_{n-1}\}$ that do not occur in S, listed in increasing order $z_1 < \cdots < z_l$. Clearly, an element of S equals e'_n, e''_n, or some previously occurring element of S exactly $l = |L|$ times. Cut S before each such element, producing l subsequences $S_1, \dots, S_l$. Append z_i to S_i to create a sequence P_i.

Now we are ready to assemble our tree T. We start by taking T to be the path consisting of edges e'_n and e''_n adjacent to a common vertex x_0. For $i = 1, \dots, l$ we form the elements of P_i into a path and affix this path to T along their (unique) common edge e. Of the two possible ways for doing this, we choose the one for which the path connecting x_0 to z_i uses this edge e.

Finally, we remove the vertex x_0, replacing e'_n and e''_n with a new edge e_n. This is the required tree $T = T(C)$.

For example, if the input is $n = 5$ and $C = (e_2, e''_5, e_2)$ (the same as in Figure 1), then $S = (e'_5, e_2, e''_5, e_2)$ and S is subdivided as

$$S_1 = (e'_5, e_2), \quad S_2 = (e''_5), \quad S_3 = (e_2).$$

Appending the remaining elements, the elements of $L = \{e_1, e_3, e_4\}$, we obtain paths

$$P_1 = (e'_5, e_2, e_1), \quad P_2 = (e''_5, e_3), \quad P_3 = (e_2, e_4),$$

which gives us back our initial tree T.

With these illustrations it is fairly clear that we indeed have a one-to-one correspondence, so we do not provide any futher details.

REFERENCES

1. P. Buneman, S. Davidson, G. Hillerbrand, and D. Suciu, A query language and optimization technique for unstructured data, in *Proc. ACM SIGMOD Int. Conf. on Management of Data* (Montreal), ACM Press, New York, 1996, pp. 505–516.
2. C. Calcagno, L. Cardelli, and A. D. Gordon, Deciding validity in a spatial logic for trees, in *ACM Workshop on Types in Language Design and Implementation* (New Orleans), ACM Press, New York, 2003, pp. 62–73.
3. P. J. Cameron, Two-graphs and trees, *Discrete Math.* **127** (1994) 63–74.
4. ———, Counting two-graphs related to trees, *Electronic J. Combin.* **2** (1995) #R4, 8 pp.

5. Ö. Eğecioğlu and J. B. Remmel, Bijection for Cayley trees, spanning trees, and their q-analogues, *J. Combin. Theory (A)* **42** (1986) 15–30.
6. D. Foata, Enumerating k-trees, *Discrete Math.* **1** (1971) 181–186.
7. J. Hage, Enumerating submultisets of multisets, *Inf. Proc. Letters* **85** (2003) 221–226.
8. J. W. Moon, Various proofs of Cayley's formula for counting trees, in *A Seminar on Graph Theory*, F. Harary, ed., Holt, New York, 1967, pp. 70–78.
9. H. Prüfer, Neuer Beweis eines Satzes über Permutationen, *Arch. Math. Phys.* **27** (1918) 142–144.

Department of Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA 15213-3890