Generating Edge-Labelled Trees

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

September 9, 2003

In this note we present a bijection enumerating *edge-labelled trees* (when we put labels on edges while vertices are indistinguishable from each another). These are not as important as vertex-labelled trees but they do appear in applications: see, for example, Cameron [3, 4], Buneman et al [1], Calgano et al [2], Hage [7].

Cameron [4, Proposition 2.1] demonstrated that the total number of edge-labelled trees with n edges is $(n+1)^{n-2}$, by showing that the number of vertex-labelled trees of size n is $n+1$ times larger than the number of edge-labelled ones.

However, it is often desirable not only to know the total number of certain objects but also to be able to generate all of them one by one. For example, one may wish to check some hypothesis for each object. In our case, it would be ideal if we could find a bijection between edge-labelled trees and X^{n-2}, where X is some fixed set of size $n+1$. This would offer extra possibilities such as sampling a uniformly distributed random edge-labelled tree by taking a random element of X^{n-2}.

There are many known bijections for vertex-labelled trees: see Prüfer [9], Moon [8], Eğecioğlu and Remmel [5], just to name a few. Combining these with Cameron's proof, one can write a bijection for edge-labelled trees but the resulting algorithm seems rather unwieldy. So Cameron [4, Problem 1] asked for a direct bijection (not going via vertex-labelled trees). Here we give one such construction, based on the ideas in Foata [6], as follows.

Let T be any tree with its edges labelled by $\{e_1, \dots, e_n\}$. Let L consist of all leaves e_i with $1 \leq i \leq n-1$. Subdivide e_n by a new vertex x_0 into two new edges e'_n and e''_n. Here we have two choices how to put these labels, so let us agree that e'_n is the edge which lies on the path from x_0 to the smallest element of L.

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

Clearly, S 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.

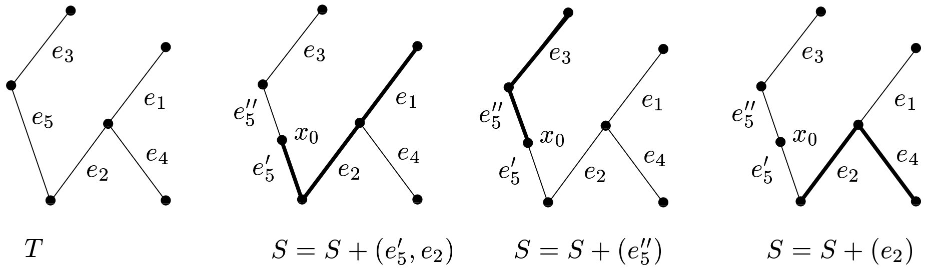

Figure 1: An illustration: Computing $C(T) = (e_2, e''_5, e_2)$

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

Now we are ready to assemble our tree T. We start with T being 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 the (unique) common edge e. Of two possible ways to do so, we choose the one so that 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 by 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 $S_1 = (e'_5, e_2)$, $S_2 = (e''_5)$ and $S_3 = (e_2)$. Appending the remaining elements $L = \{e_1, e_3, e_4\}$, we obtain paths $P_1 = (e'_5, e_2, e_1)$, $P_2 = (e''_5, e_3)$ and $P_3 = (e_2, e_4)$, 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 optimazation technique for unstructured data*, Proc. ACM SIGMOD Int. Conf. on Management of Data, ACM press, 1996, pp. 505–516.

[2] C. Calcagno, L. Cardelli, and A. D. Gordon, *Deciding validity in a spatial logic for trees*, ACM Workshop on Types in Language Design and Implementation, New Orleans, ACM Press, 2003, pp. 62–73.

[3] P. J. Cameron, *Two-graphs and trees*, Discrete Math. **127** (1994), 63–74.

[4] P. J. Cameron, *Counting two-graphs related to trees*, Electronic J. Combin. **2** (1995), no. #R4, 8pp.

[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 Caley's formula for counting trees*, 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.