

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 .

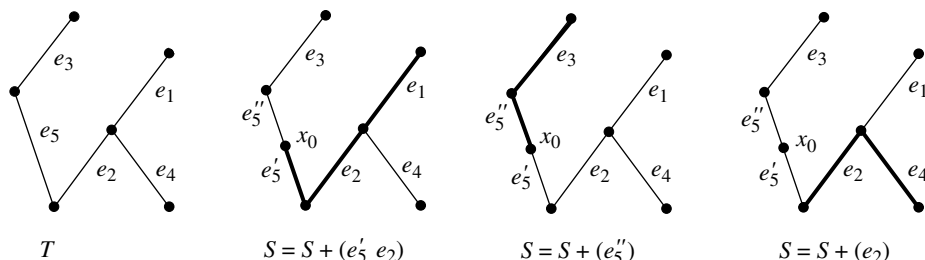


Figure 1. An illustration: computing $C(T) = (e_2, e'_5, e_2)$.

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 $z_1 < \dots < z_l$. 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 further 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