

# Online Suffix Trees with Counts\*

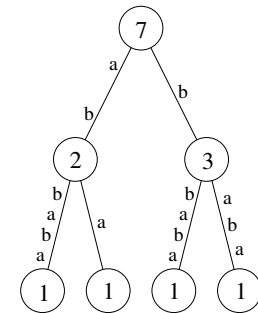
Breannán Ó Nualláin<sup>†</sup>

Steven de Rooij<sup>‡</sup>

We extend Ukkonen's online suffix tree construction algorithm to support substring frequency queries, by adding count fields to the internal nodes of the tree. This has applications in the field of sequential data compression, for example in the implementation of an efficient PPM-style compression algorithm with unbounded context length. Slightly differently than suggested in [1], we let the count field of a node invariantly equal the number of occurrences of the concatenation of the consecutive edge labels going from the root to that node. We show that under this invariant, due to the onlineness requirement, the algorithm's worst case time complexity is  $O(n^2)$ . However, its average case performance is  $\sim n \log n$  under reasonable assumptions, so it may well be an acceptable solution in practice.

One major problem that we address is the fact that Ukkonen's online construction algorithm does not maintain explicit end of string markers in the tree. Consider a suffix tree that corresponds to a string  $T$ . If we want to know the number of occurrences of a string  $u$  in  $T$ , we have to follow a path from the root such that the consecutive edge labels spell out  $u$ . Since the tree is path compressed (compact), we may end up somewhere halfway along some edge. Letting  $v$  be the remainder of the edge label, we can find the number of occurrences of  $uv$  by looking at the count field of the node that corresponds to  $uv$ . We know that no copy of  $u$  in  $T$  has ever been followed by something other than  $v$ , or there would be a corresponding branching point in the tree. But since there are no end markers in the tree, there may be a proper prefix  $w$  of  $v$  such that  $uw$  is a suffix of  $T$ , in which case  $u$  occurred more often than  $uv$ .

It is impractical to add end markers to the tree, since an online algorithm would have to add and remove them again every time the tree is updated with a new symbol. The major part of our work concerns quickly determining where the end markers for a particular edge would be, so that frequencies can be correctly obtained. We give a complete characterization of all end markers on leaf edges. Furthermore we found that edges between two internal nodes can contain at most one end marker. Using these results we give algorithms to update the count fields and do frequency queries correctly. All algorithms have been implemented and tested correct in practice.



Tree for  
 $T = \text{"abbaba"}$ .  
How often did "a"  
occur in  $T$ ?

## References

- [1] N. Jesper Larsson. *Structures of String Matching and Data Compression*. Ph.D. thesis, Department of Computer Science, Lund University, Sweden, September 1999.
- [2] Steven de Rooij. *Methods of Statistical Data Compression*. Master's thesis, University of Amsterdam, September 2003. <http://www.cwi.nl/~rooij>.

\*Much of this material first appeared in [2].

<sup>†</sup>Affiliation: Fac. of Science, University of Amsterdam, The Netherlands. Email: bon@science.uva.nl.

<sup>‡</sup>Corresponding author. Affiliation: Centrum voor Wiskunde en Informatica (CWI), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands. Phone: +31 20 5929333; Fax: +31 20 5924199. Email: rooij@cwi.nl.