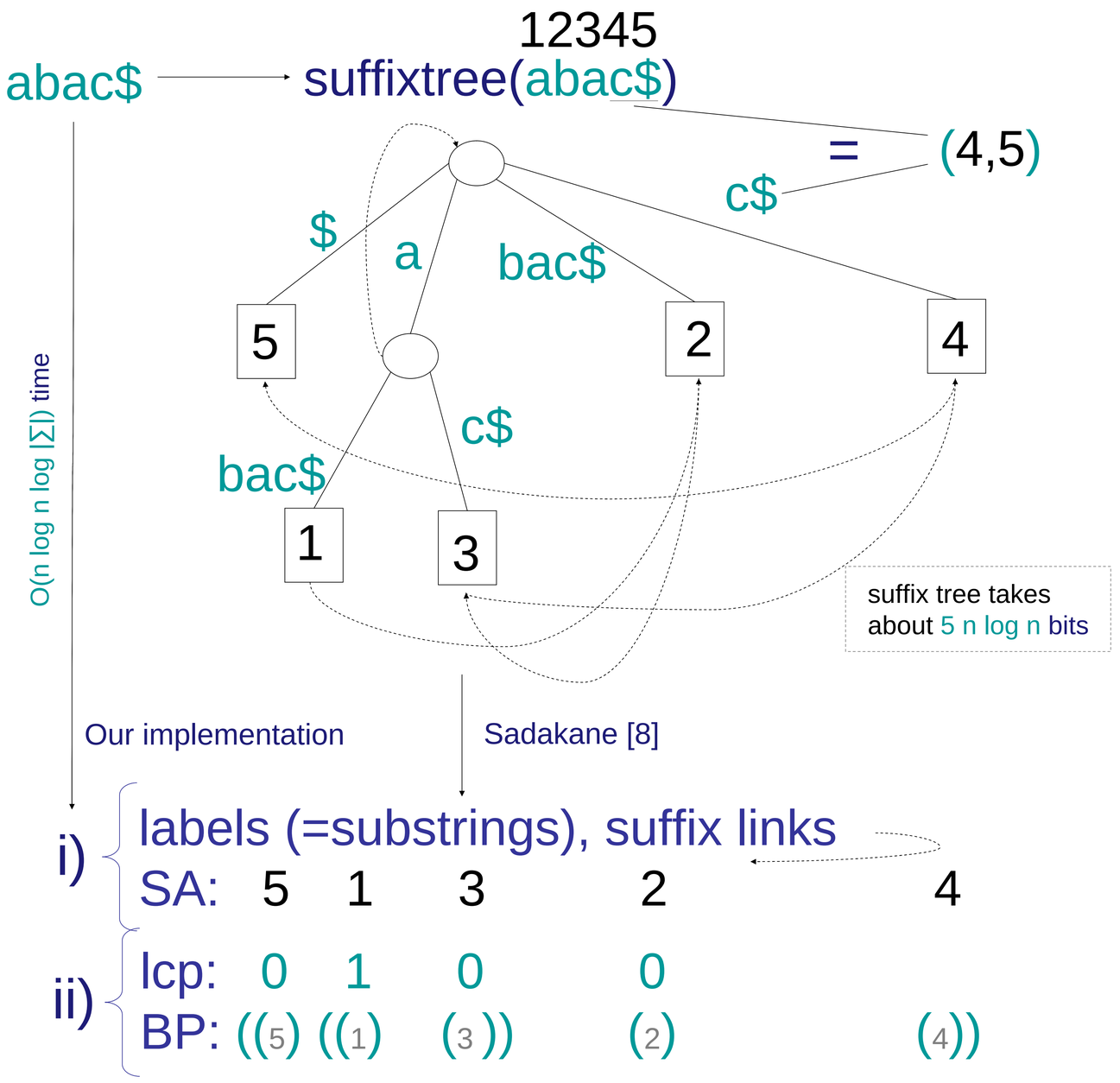
HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Suffix tree is probably the most important data structure when it comes to analyse strings. It finds applications in sequence comparison, pattern discovery, string algorithms, information retrieval, etc. It is hard to find a non-trivial question on strings whose efficient solution wouldn't require a suffix tree to be build. The bottleneck in the practical use of suffix trees has been its space-greedyness. Recently, *compressed suffix trees* have been proposed that work in significantly smaller space, and support all typical suffix tree operations with a slight slowdown. We offer the first implementation of such structure [8] together with a space-economical construction algorithm.

Implementation of Compressed Suffix Trees



i) *FM-index* [2,3] is a compressed variant of *suffix array* (SA) supporting O(|P|) time algorithm to count how many times a pattern P occurs in a text. Main incredience of FM-index is *Burrows-Wheeler* (BW) transform [1].

It is possible to construct a *dynamic FM-index* using no more space than the compressed index itself [7]. This provides us with the BW-transform.

Sadakane shows that suffix links from leaves and labels of edges can be computed by means of *compressed suffix arrays* (CSA) that support functions SA[i], SA^{-1}[i], substring(i,j).

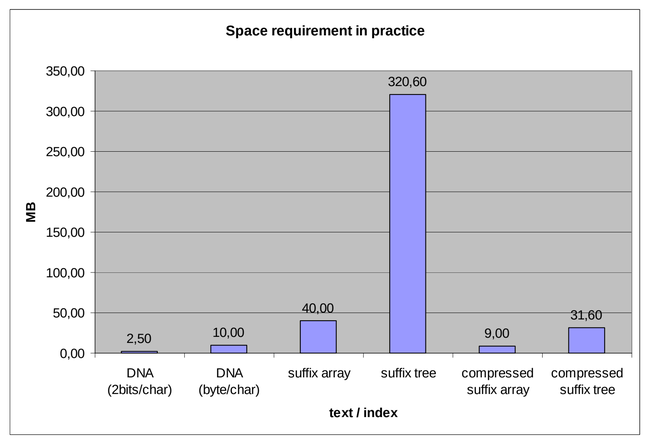
Our result: We show that, given BW-transform, we can construct the above CSA using no additional space.

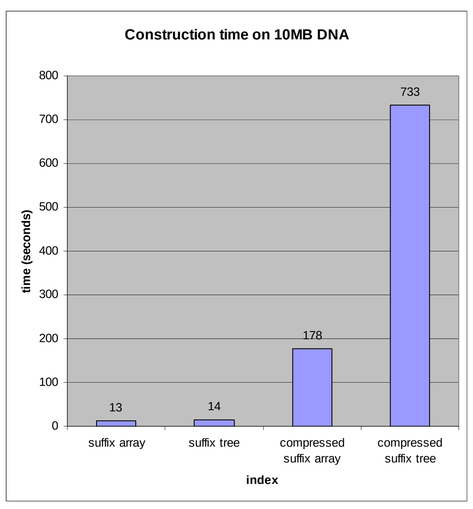
Suffix links from internal nodes can be computed using *lowest common ancestor* (lca) queries. Space-efficient structures for constant time lca-queries were studied by Sadakane. We engineered a practical version by adding a space/time tradeoff.

ii) Sadakane shows that lcp values can be stored in 2n+o(n) bits so that each value can be computed in constant time. We modified Kasai et al. [5] algorithm to directly construct the compressed lcp values in linear time.

Munro et al. [6] show that, using 4n bits balanced parantheses (BP) representation of the tree together with small data structures taking o(n) bits, one can simulate tree traversals in constant time per step. We modified related earlier implementations to solve these tasks.

We developed an algorithm to construct BP representation by scanning (compressed) lcp values from left to right, maintaining the right-most path of the tree using only O(n) bits extra space. Similar algorithm has also been proposed by Hon and Sadakane [4].





[1] M. Burrows and D. Wheeler. A block sorting lossless data compression algorithm. Techical report 124, DEC, 1994.

[2] P. Ferragina and G. Manzini. Indexing compressed texts. *JACM*, 52(4):552-581, 2005.

[3] P. Ferragina, G. Manzini, V. Mäkinen, and G. Navarro. Compressed representattions of sequences and full-text indexes. *ACM TALG*, in press.

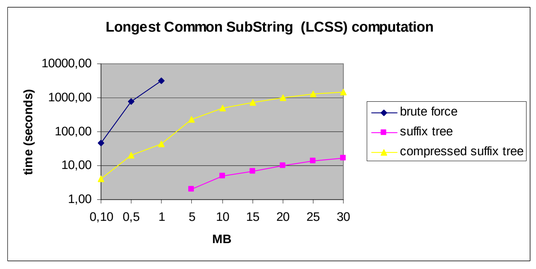
[4] W.-K. Hon and K. Sadakane. Space-economical algorithms for finding maximum unique matches. In *Proc. CPM 2002*, pp. 133-152.

[5] T. Kasai et al. Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Proc. CPM 2001*, pp. 181-192.

[6] I. Munro, V. Raman, and S. Rao. Space efficient suffix trees. *JALG*, 39(2):205-222, 2001.

[7] V. Mäkinen and G. Navarro. Dynamic entropy-compressed sequences and full-text indexes. In *Proc. CPM 2006*, pp. 306-307.

[8] K. Sadakane. Compressed suffix trees with full functionality, *Theory of Computing Systems*, to appear.



Implementation available at http://www.cs.helsinki.fi/group/suds

Department of Computer Science

Faculty of Science

Kashyap Dixit, Wolfgang Gerlach, Veli Mäkinen, and Niko Välimäki