Entropy-compressed suffix trees

Given a text $T[1, n]$ over an alphabet of size s, a plain representation requires $n \log s$ bits (log is to base 2). A k-th order compressor can reduce its size to nHk bits, where Hk is the empirical k-th order entropy of T [1]. Classical text indexes such as suffix trees and suffix arrays [2] require $O(n \log n)$ bits. This waste of space is troublesome when indexing large texts that could fit in main memory but whose indexes (sometimes 20 times larger than the text!) cannot. Thus there is a strong interest in reduced-space representations that retain reasonable efficiency.

Many recent developments [3] achieve suffix array functionality using $nHk + o(n \log s)$ bits, for any $k \leq a \log_s n$ and any constant $0 < a < 1$. This space also contains the text, in the sense that the structure is capable of reproducing any text substring. By "suffix array functionality" I mean counting the number of occurrences of any pattern, and enumerating its text positions. The former can be done, say, in $O(m \log s)$ time (m being the pattern length), and the latter in $O(\text{polylog}(n))$ time per reported occurrence.

Suffix tree functionality is more ambitious. It permits navigating the (concrete or virtual) suffix tree with operations like parent, child-labeled-a, first-child, next-sibling, suffix-link (leading from node representing ax to node representing x, a being a symbol and x a string), queries like subtree-size, first-leaf, last-leaf, and optionally other more ambitious ones like level-ancestor, lowest-common-ancestor, etc.

There have been recent achievements on succinct suffix trees with full functionality, most notably Sadakane's [4]. Yet all of them still require $O(n)$ extra bits of space on top of the entropy. In principle, $nHk + o(\ldots)$ bits should be sufficient (as at worst one can uncompress the text, build the suffix tree, and do the operation!), but no one has devised a way to operate efficiently on a suffix tree structure that is fully entropy-compressed, without any extra linear space. I believe this should be possible.

References

1. G. Manzini. An analysis of the Burrows-Wheeler transform. Journal of the ACM 48(3):407-430, 2001.
2. D. Gusfield. Algorithms on strings, trees, and sequences: Computer Science and Computational Biology. Cambridge University Press, 1997.
3. G. Navarro and V. Makinen. Compressed full-text indexes. ACM Computing Surveys 39(1):article 2, 2007.
4. K. Sadakane. Compressed suffix trees with full functionality. Theory of Computing Systems, to appear. Preliminary version available at http://tcslab.csce.kyushu-u.ac.jp/~sada/papers/cst.ps

Gonzalo Navarro,
University of Chile,
Chile.