Bachelor's Thesis

Compressed Bit Vectors with Rank and Select Support

Overview

Many compact and succinct data structures are based on *bit vectors*. They can be used to represent a tree with n nodes in $2n + o(n)$ bits, e.g., [5]. The $o(n)$ bits allow us to fully navigate within the tree. Other applications for bit vectors are wavelet trees with their myriads of different use-cases [1, 4, 8, 9]. Here, we reorder the bit representations of a text to better answer rank and select queries, i.e., answering the questions: "how often a character exists up to some position?" and "where a character occurs the j-th time?" For both applications, rank and select queries have to be answered on the bit vector.

Another notable application is the Elias-Fano encoding [2, 3] of monotonically increasing functions. Given n monotonic increasing integers from a universe of size u, the sequence can be encoded using only $n(2 + \log(u/n)) + 1 + o(n)$ bits, such that not only access but also predecessor and successor queries can be answered.

This encoding plays an important role in PaCHash, a packed and compressed hash table [7]. Here, we make use of a very space efficient rank and select implementation for bit vectors [6]. This implementation has a space-overhead of only $3.51\,\%$. To achieve this low space-overhead, there are multiple levels of indices that store the number of ones using as little bits as possible, cf. Fig. 1.

Objective

The main objective of this Bachelor's thesis is to develop a compression for the bit vector, such that rank and select queries still work (reasonable) fast. To this end, the bit vector should be compressed in such a way that it can be decompressed on-the-fly.

An optional goal of this thesis is the development of an encoding specifically for the bit vectors produced by PaCHash. These bit vectors have some properties which may be used to compress them even better than when using entropy encoding.

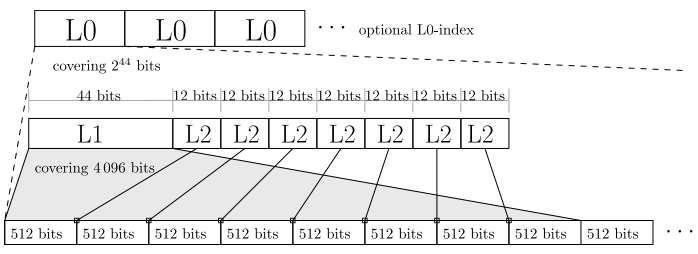

Figure 1: Schematic of the rank data structure [6] used by PaCHash [7].

Requirements

- Excellent C++ programming skills
- Interest in compact and compressed data structures

Contact

Dr. Florian Kurpicz (kurpicz@kit.edu)

References

[1] Patrick Dinklage, Jonas Ellert, Johannes Fischer, Florian Kurpicz, and Marvin Löbel. Practical wavelet tree construction. *ACM J. Exp. Algorithmics*, 26:1.8:1–1.8:67, 2021.

[2] Peter Elias. Efficient storage and retrieval by content and address of static files. *J. ACM*, 21(2):246–260, 1974.

[3] Robert Mario Fano. On the number of bits required to implement an associative memory, 1971.

[4] Paolo Ferragina, Raffaele Giancarlo, and Giovanni Manzini. The myriad virtues of wavelet trees. *Inf. Comput.*, 207(8):849–866, 2009.

[5] Guy Jacobson. Space-efficient static trees and graphs. In *FOCS*, pages 549–554. IEEE Computer Society, 1989.

[6] Florian Kurpicz. Engineering compact data structures for rank and select queries on bit vectors. *Accepted at SPIRE*, 2022.

[7] Florian Kurpicz, Hans-Peter Lehmann, and Peter Sanders. Pachash: Packed and compressed hash tables. *CoRR*, abs/2205.04745, 2022.

[8] Christos Makris. Wavelet trees: A survey. *Comput. Sci. Inf. Syst.*, 9(2):585–625, 2012.

[9] Gonzalo Navarro. Wavelet trees for all. *J. Discrete Algorithms*, 25:2–20, 2014.