Overview

The FM-index combines two data structures: the Burrows-Wheeler transform and wavelet trees. It is a very prominent full-text index and used in most DNA read aligners [4] and in Bioinformatics in general. In this Master’s thesis, we focus on the efficient construction of the second data structure—wavelet trees.

The wavelet tree is a binary tree data structure that can be used to answer rank and select queries on texts of size $n$ over an alphabet of size $\sigma$ in $O(\lg \sigma)$ time. Here, $\text{rank}_\sigma(i)$ queries ask for the number of occurrences of the symbol $\alpha$ before the position $i$ and $\text{select}_\sigma(i)$ queries return the text position of the $i$-th occurrence of the symbol $\alpha$.

Let $T$ be a text of length $n$ over an alphabet of size $\sigma$. The corresponding wavelet tree consists of $\lceil \lg \sigma \rceil$ bit vectors of size $n$, see Fig. 1. Even though all $n \lceil \lg \sigma \rceil$ entries in the bit vectors have to be looked at during construction, the wavelet tree can be computed in $O(n \lg \sigma / \sqrt{\ln n})$ time using broadword programming [1, 5]. There exists an implementation of such an algorithm by Kaneta [3], which heavily relies on specialized CPU instructions like parallel bit extract and packed shuffle bytes. The reported construction times are faster than the previously fastest sequential WT construction algorithm [2]. However, the algorithm has one significant disadvantage—it requires twice as much memory.

Objective

The main objective of this Master’s thesis is to develop a fast and space efficient wavelet tree construction algorithm that computes the wavelet tree in $O(n \lg \sigma / \sqrt{\ln n})$ time using specialized CPU instructions.

The parallelization of this algorithm is another (minor) goal of this Master’s thesis. There exists a very fast meta-algorithm that can be used to parallelize all wavelet tree construction algorithms, e.g., [2] and it should be experimentally evaluated whether a parallelization of the algorithm described above is faster than using the meta-algorithm for the parallelization.

Requirements

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

Contact

Florian Kurpicz (kurpicz@kit.edu)