Algorithms for the compression of genomic big data

Nicola Prezza¹ and Alberto Policriti¹,²

1University of Udine, Department of Mathematics, Physics, and Computer Science
2Institute of Applied Genomics, Udine

Abstract

Motivations Building the Burrows-Wheeler transform (BWT) and computing the Lempel-Ziv parsing (LZ77) of huge collections of genomes is becoming an important task in bioinformatic analyses as these datasets often need to be compressed and indexed prior to analysis. Given that the sizes of such datasets often exceed RAM capacity of common machines however, standard algorithms cannot be used to solve this problem as they require a working space at least linear in the input size. One way to solve this problem is to exploit the intrinsic compressibility of such datasets: two genomes from the same species share most of their information (often more than 99%), so families of genomes can be considerably compressed. A solution to the above problem could therefore be that of designing algorithms working in compressed working space, i.e. algorithms that stream the input from disk and require in RAM a space that is proportional to the size of the compressed text.

Methods In this work we present algorithms and data structures to compress and index text in compressed working space. These results build upon compressed dynamic data structures, a sub-field of compressed data structures research that is lately receiving a lot of attention. We focus on two measures of compressibility: the empirical entropy $H$ of the text and the number $r$ of equal-letter runs in the BWT of the text. We show how to build the BWT and LZ77 using only $O(Hn)$ and $(r \log n)$ working space, $n$ being the size of the collection. For the case of repetitive text collections (such as sets of genomes from the same species), this considerably improves the working space required by state-of-the art algorithms in the literature. The algorithms and data structures here discussed have all been implemented in a public C++ library, available at github.com/nicolaprezza/DYNAMIC. The library includes dynamic gap-encoded bitvectors, run-length encoded (RLE) strings, and RLE FM-indexes.

Results We conclude with an overview of the experimental results that we obtained running our algorithms on highly repetitive genomic datasets. As expected, our solutions require only a small fraction of the working space used by solutions working in non-compressed space, making it feasible to compute BWT and LZ77 of huge collections of genomes even on desktop computers with small amounts of RAM available. As a downside of using complex dynamic data structures however, running times are still not practical so improvements such as parallelization may be needed in order to make these solutions fully practical.