Compror: compression with a factor oracle

Arnaud Lefebvre – Thierry Lecroq
UMR CNRS 6037-ABISS Université de Rouen, 76821 Mt St-Aignan Cedex, France
{arnaud.lefebvre,thierry.lecroq}@univ-rouen.fr

Compror is a text compression method using a factor oracle [1]. The factor oracle of a word \(x\) is a space economical index structure which represents at least all the factors of \(x\). It is an automaton which representation, in addition to \(x\), consists in only a small number of transitions (see 1). It can be constructed in linear time and space. It is also possible to compute, with the same complexities, a position and a length of a long repeated suffix for each position of \(x\) [2]. This enables to compute a factorization of \(x\). This factorization is used during the encoding process as follows: \(x = uv\) where the prefix \(u\) of \(x\) as already been encoded and the suffix \(v\) has to be encoded. The current state in the oracle is \(q\), corresponding to \([u]\). Then the characters of \(v\) are parsed through the oracle, from state \(q\), as long as the length of the repeated suffix is greater than the prefix of \(v\) being processed. Each time a new letter is encountered, it is encoded as a letter, otherwise a repeated segment of \(x\) is encoded as a pair (starting position,length). For instance, the word \(aabbabbaabab\) is encoded by \((1,1)b(3,1)(2,8)\).

![Figure 1: Factor oracle of aabbabbaabab.](image)

Figure 1: Factor oracle of \(aabbabbaabab\). Plain arrows represent the transitions. Values near the states are the lengths of a repeated suffix, its ending positions are indicated with the dashed arrows.

The construction of the oracle and the encoding process are done simultaneously. Fibonacci codes of orders 2 and 3 are actually used during this phase. The decoding process is straightforward. This on-line lossless compression method is easy to compute and can handle very large texts. The compression ratios are worst than the ones of gzip or bzip2, but the compression and decompression times are similar to the ones of bzip2.

References


*This work was partially supported by a NATO grant PST.CLG.977017