Fast Decoding of Fibonacci Encoded Texts

Shmuel T. Klein
Department of Computer Science
Bar Ilan University, Ramat-Gan 52900, Israel
Tel: (972–3) 531 8865 Fax: (972–3) 736 0498 tomi@cs.biu.ac.il

The Fibonacci code of order \( m \geq 2 \), denoted by Fib\(_m\), is defined as the (infinite) set of binary strings \( w \), each of which is terminated by \( m \) 1s, and such that a substring of \( m \) consecutive 1s does not appear anywhere in \( w \) except at its right end. For example, for \( m = 2 \), the shortest codewords of Fib2 are 11, 011, 0011, 1011, 00011, 01011, etc. The major advantage of the Fib\(_m\) codes is the fact that the codeword set is fixed, which simplifies both encoding and decoding. The loss incurred by not using an optimal (Huffman) code is often tolerable, and other non-optimal variants with desirable features, such as faster processing and simplicity have been suggested, for example Tagged Huffman codes, End-Tagged Dense codes and \((s, c)\)-Dense codes.

These three last mentioned codes are byte-aligned, allowing very fast processing, whereas the Fib\(_m\) codes have codewords of any length \( \geq m \) in bits. This work deals with ways to accelerate the processing, concentrating on the decoding of Fibonacci encoded texts, by adapting a method originally suggested for the fast decoding of Huffman codes, which suffer from the same problem as the variable length Fib\(_m\) codes. The method uses a set of partial decoding tables that are prepared in advance and depend only on the code, not on the actual text to be decoded, by means of which the decoding is then performed by blocks of \( k \) bits at a time, rather than bit per bit. Typically, \( k = 8 \) or 16. One therefore gets a time/space tradeoff, where faster decoding comes at the cost of storing larger tables.

The main problem with this partial decoding approach is that the number of required tables and their sizes may be prohibitively large, but it is possible to reduce the number of required tables using the properties of the Fibonacci numeration systems on which the Fib\(_m\) codes are based. Contrarily to Huffman codes, for which a substring of a codeword has no special relation with the full codeword it has been extracted from, in the Fibonacci case, the codewords are representations of integers in a Fibonacci numeration system, so a codeword can be expressed as a suitable combination of its subparts. Therefore, a codeword split by the boundary of a \( k \)-bit block can be reconstructed efficiently.

The proposed decoding algorithm performs only a small number of fast operations (table access, additions) per \( k \)-bit block. The overall required storage overhead with \( k = 8 \) for alphabets of size \( N = 10,000, 100,000 \) and \( 1,000,000 \) is only 10K, 15K and 21.4K, respectively. For \( k = 16 \), with twice as fast decoding, the overhead would be 1.9M, 2.6M and 3.4M, respectively. The original decoding using the full set of tables would require, for \( N = 100,000 \), at least 300M with \( k = 8 \), and 75G with \( k = 16 \), which would actually result in a slowdown rather than a speedup because of page faults and cache misses.