Space-Economical Generation of Minimum Redundancy Codes

Artur Alves Pessoa, Ruy Luiz Milidiu, Eduardo Sany Laber.
Deparamento de Informática, PUC-RJ, Brasil
{artur,milidiu,laber}@inf.puc-rio.br

Given an alphabet $\Sigma = \{a_1, \ldots, a_n\}$ with an associated list of weights $W = [w_1, \ldots, w_n]$, an optimal prefix code for Σ is a prefix code that minimizes $\sum_{i=1}^{n} w_i l_i$, where l_i is the length of the code assigned to symbol a_i. Minimum-redundancy coding plays an important role in data compression applications. For an input list of n symbol weights, the well-known Huffman's algorithm generates a corresponding set of minimum-redundancy codewords in $O(n \log n)$ time and $O(n)$ extra space. The time complexity of Huffman's algorithm can be reduced to $O(n)$ if the input list is already sorted.

Moffat and Turpin [1] proposed a space-efficient representation of minimum-redundancy codes: a list $M = [m_1, \ldots, m_H]$ of H *runlengths*, where H is the length of the longest codeword. For $i = 1, \ldots, H$, the value of m_i is defined as the number of codewords with length i in the resulting code. It is worth to mention that, from this output, the translation table for a corresponding Huffman-Shannon-Fano code is easily constructed in $O(H)$ time and using just $O(H)$ space.

Here, we introduce a lazy approach to traverse a Huffman tree constructed for an input list of n sorted weights in $O(n)$ time and using only $O(H)$ additional space. Based on the resulting method, that we call the *Lazy Traverse* algorithm, we present three algorithms for calculating minimum-redundancy codes, that we respectively call Fast LazyHuff (F-LazyHuff), Economical LazyHuff (E-LazyHuff) and LazyHuff. F-LazyHuff is the simplest one. It executes in $O(n)$ time and requires $O(\min\{H^2, n\})$ additional space. On the other hand, E-LazyHuff needs only $O(H)$ additional space, spending however $O(n \log(n/H))$ time to execute. Finally, LazyHuff is a refined version of E-LazyHuff, that executes in $O(n)$ time and requires only $O(H)$ additional space.

[1] Moffat, A. and Turpin, A. *Efficient construction of minimum-redundancy codes for large alphabets*, **IEEE Transactions on Information Theory, 4, vol 44, 1998.**