Research Proposal:

Adaptive Analysis of Optimal Prefix Free Codes' Computation

Jérémy Barbay and David Muñoz

Departmento de Ciencias de la Computación,
University of Chile,
jeremy@barbay.cl,
david.munoz@ug.uchile.cl

Abstract. We propose to study algorithms computing optimal prefix free codes from N unordered positive integer weights in the comparison model, so that to improve on large family of instances over the state of the art complexities of $O(N \lg N)$ in the algebraic decision tree model for the computation of Huffman's codes, a landmark in compression and coding since 1952.

Keywords: Counting Sort, Huffman Code, Minimal Redundancy, Optimal Prefix Free Code.

Definition of the Problem

Given N positive integer weights $W[1..N]$ coding for the frequencies $\left\{W[i]/\sum_{j=1}^{N} W[j]\right\}_{i\in[1..N]}$ of N messages, and a number D of output symbols, an *Optimal Prefix Free Code* [7] is a set of N code strings on alphabet $[1..D]$, of variable lengths $L[1..N]$ and such that no string is prefix of another, and the average length of a code is minimized (i.e. $\sum_{i=1}^{N} L[i]W[i]$ is minimal). The particularity of such codes is that even though the code strings assigned to the messages can differ in lengths (assigning shorter ones to more frequent messages yields compression to $\sum_{i=1}^{N} L[i]W[i]$ output symbols), the prefix free property insures a non-ambiguous decoding. Such optimal codes, known since 1952 [7], are used in "*all the mainstream compression formats*" [3] (e.g. PNG, JPEG, MP3, MPEG, GZIP and PKZIP). The concept is "*one of the fundamental ideas that people in computer science and data communications are using all the time*" (Knuth [13]), and "*one of the enduring techniques of data compression. It was used in the venerable PACK compression program, authored by Szymanski in 1978, and remains no less popular today*" (Moffat *et al.* [11] in 1997).

Existing Solutions

Any prefix free code can be computed in linear time from a set of code lengths satisfying the Kraft inequality $\sum_{i=1}^{N} D^{-L[i]} \leq 1$. The original description of the code by Huffman [7] yields a heap-based algorithm performing $\mathcal{O}(N \log N)$ algebraic operations, using the bijection between D-ary prefix free codes and D-ary cardinal trees. This complexity is asymptotically optimal for any constant value of D in the algebraic decision tree model, in the worst case over instances composed of N positive integer weights, as computing the optimal binary prefix free code for the weights $W[0,\ldots,DN] = \{D^{x_1},\ldots,D^{x_1},D^{x_2},\ldots,D^{x_2},\ldots,D^{x_N},\ldots,D^{x_N}\}$ is equivalent to sorting the positive integers $\{x_1,\ldots,x_N\}$.

Two types of results go beyond this lower bound on the computational complexity of algebraic algorithms computing binary prefix free codes:

1. van Leeuwen described a reduction in $\mathcal{O}(N)$ algebraic operations [8] to sorting the weights (i.e. the converse to the lower bound described above), which yields solutions of complexity $\mathcal{O}(N \log\log N)$ in the worst case [4, 5] and $\mathcal{O}(N\sqrt{\log\log N})$ in the average case [6] when sorting the weights using non algebraic operations (see Table 1 for a partial list of results from sorted weights).

2. Belal *et al.* [2] described an algorithm claimed to perform $\mathcal{O}(kN)$ algebraic operations for the unsorted case, in the worst case over instances formed by N weights and for which there is an optimal binary prefix free code with k distinct code lengths[1] (see Table 2 for a partial list of results from unsorted weights).

Year	Name	Time	Space	Ref.	Note
1976	van Leeuwen	$O(N)$	$O(N)$	[8]	
1995	Moffat and Katajainen	$O(N)$	$O(1)$	[10]	
1998	Moffat and Turpin	$O(r(1+\log(N/r)))$	"space efficient"	[12]	outputs run lengths
2001	Fast Lazy Huffman	$O(N)$	$O(\min\{H^2, N\})$	[9]	outputs run lengths
2001	Economic Lazy Huffman	$O(N(1+\log(N/H)))$	$O(H)$	[9]	outputs run lengths
2001	Best Lazy Huffman	$O(N)$	$O(H)$	[9]	outputs run lengths
2006	Belal and Elmasry	$O(\log^{2k-1} N)$		[2]	proof not clear

Table 1. A selection of results on the computational complexity of optimal prefix free codes from sorted weights. H denotes the lenght of the longest prefix free code produced.

Year	Name	Time	Space	Ref.	Note
1952	Huffman	$O(N \log N)$	$O(N)$	[7]	original
2002	Han	$O(N \lg\lg N)$	$O(N)$	[4, 5]	integer sorting
2006	Belal and Elmasry	$O(kN)$ claimed	$O(N)$	[2]	k distinct code lengths
2006	Belal and Elmasry	$O(16^k N)$ proved	$O(N)$	[1]	k distinct code lengths
2012	Integer Sorting	$O(N\sqrt{\lg\lg N})$ average	$O(N)$	[6]	randomized int sorting

Table 2. A selection of results on the computational complexity of optimal prefix free codes from unsorted weights. k is the number of distinct codelenghts produced.

Note that there can be various optimal codes for any given set of weights, each with distinct maximal code length H or number of distinct code lengths k

Hypothesis

Given N positive integer weights, can we compute an optimal binary prefix free code in time better than

1. $\mathcal{O}(N \log N)$ for some large classes of instances, even when all the weights are distinct, or better than
2. $O(16^k N)$ when the code computed by Huffman's solution has k distinct code lengths?

Objectives

The objectives of this internship will be three-fold:

1. Survey the existing solutions to compute optimal prefix free codes;
2. Define a "certificate" data structure for the computation of an optimal prefix free codes, which can be quickly checked when a small subset of the weights change, in order to assert the optimality of the prefix free code previously returned on the new instance;
3. Define a fine analysis of the computation of optimal prefix free code based on the size of this certificate, knowing that this size will be so large in the worst case as to force a complexity within $\mathcal{O}(N \log N)$, but with the hope that this size is small in many other cases, and yields faster algorithms on such instances.

[1] Note that k is not uniquely defined, as for a given set of weights there can exist several optimal prefix free codes varying in the number of distinct code lengths used.

Bibliography

[1] A. A. Belal and A. Elmasry. Distribution-sensitive construction of minimum-redundancy prefix codes. *CoRR*, abs/cs/0509015, 2005. Version of Tue, 21 Dec 2010 14:22:41 GMT, with downgraded results from the ones in the conference version [2].

[2] A. A. Belal and A. Elmasry. Distribution-sensitive construction of minimum-redundancy prefix codes. In B. Durand and W. Thomas, editors, *Proceedings of the International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 3884 of *Lecture Notes in Computer Science*, pages 92–103. Springer, 2006.

[3] C. Chen, Y. Pai, and S. Ruan. Low power Huffman coding for high performance data transmission. In *Hybrid Information Technology, 2006. ICHIT'06. International Conference on*, volume 1, pages 71–77. IEEE, 2006.

[4] Y. Han. Deterministic sorting in $O(n \log\log n)$ time and linear space. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 602–608, New York, NY, USA, 2002. ACM.

[5] Y. Han. Deterministic sorting in $o(n \log\log n)$ time and linear space. *Journal of Algorithms*, 50(1):96 – 105, 2004.

[6] Y. Han and M. Thorup. Integer sorting in $O(n\sqrt{\log\log n})$ expected time and linear space. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 135–144, 2002.

[7] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, September 1952.

[8] J. V. Leeuwen. On the construction of Huffman trees. In *Proceedings of the International Conference on Automata, Languages, and Programming (ICALP)*, pages 382–410, Edinburgh University, 1976.

[9] R. L. Milidiú, A. A. Pessoa, and E. S. Laber. Three space-economical algorithms for calculating minimum-redundancy prefix codes. *IEEE Trans. Inf. Theor.*, 47(6):2185–2198, Sept. 2001.

[10] A. Moffat and J. Katajainen. In-place calculation of minimum-redundancy codes. In *WADS*, volume 955 of *Lecture Notes in Computer Science*, pages 393–402. Springer, 1995.

[11] A. Moffat and A. Turpin. On the implementation of minimum redundancy prefix codes. *IEEE Transactions on Communications*, 45(10):1200–1207, 1997.

[12] A. Moffat and A. Turpin. Efficient construction of minimum-redundancy codes for large alphabets. *IEEE Transactions on Information Theory*, pages 1650–1657, 1998.

[13] M. N. Chandrasekaran. *Discrete Mathematics*. PHI Learning Pvt. Ltd., 2010.