Simple Worst-Case Optimal Adaptive Prefix-Free Coding

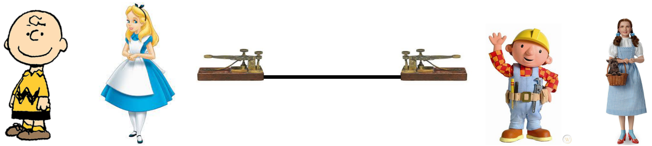
Travis Gagie
Dalhousie University

DCC 2022

Alice, Bob, Charlie and Dorothy

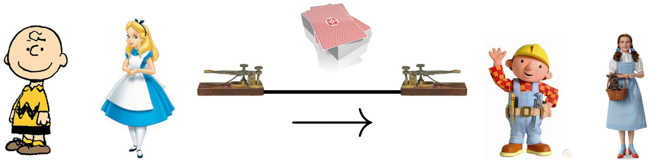
Alice lives in Vancouver, Bob lives in Halifax and they make a living operating a noiseless binary channel between those cities, charging 1¢ per bit transmitted.

Charlie in Vancouver and Dorothy in Halifax are Alice and Bob's most challenging clients, always asking for odd things and fretting about the cost or the wait.



Huffman coding

One day, Charlie comes to Alice with a deck of n cards, each with a character from an alphabet of size $\sigma \ll n$. He wants to shuffle the deck, then draw the cards one by one and have Alice send each character to Dorothy via Bob before Charlie draws the next card. He asks what bounds Alice can give on what it will cost and how long it will take.



Huffman coding

Alice asks if she can examine the deck before they start and Charlie agrees. She says she can then use

$$n(H + \delta) + o(n)$$

bits, where H is the entropy of the normalized distribution of characters in the deck and δ is the redundancy of a Huffman code for that distribution.

Huffman coding

If Alice and Bob use a canonical Huffman code, then she can use $O(n)$ total time to encode all the characters and he'll use $O(n \log \log n)$ total time to decode them.

Charlie thinks it over, agrees again, and they make the transmission.

length-restricted Huffman coding

The next day, Charlie says Dorothy was annoyed at having to wait $O(n \log\log n)$ time while he waited only $O(n)$ time. He asks if she could wait $O(n)$ time as well.

Alice remembers a result by Milidiú and Laber that says L-restricting a Huffman code increases the redundancy by less than $1/\varphi^{L - \lceil \lg(\sigma + \lceil \lg \sigma \rceil - L) \rceil - 1}$.

length-restricted Huffman coding

Alice realizes that if she and Bob restrict their codewords to length $\lceil \lg \sigma + \lg \lg n \rceil$ then the bound on the length of the encoding increases by only $O\left(n / \varphi^{\lg \lg n}\right) \subset o(n)$.

Because the longest codeword has at most $\lceil \lg \sigma + \lg \lg n \rceil$ bits, Bob can build an $O(\sigma \log n)$-space lookup table that lets him decode all the characters in $O(n)$ total time.

Charlie thinks that's neat, so they make another transmission.

Simple Optimal Adaptive Prefix Coding

Travis Gagie

adaptive coding

On the third day, Charlie has a new deck of n cards over the same alphabet, and for some reason he doesn't want Alice to examine the deck before they start.

adaptive coding

Alice says the easiest options are

- use the Faller-Gallager-Knuth algorithm and at most $n(H + 2 + \delta) + o(n)$ bits
- use Vitter's algorithm and at most $n(H + 1 + \delta) + o(n)$ bits
- use Gagie's modification of FGK for adaptive Shannon coding and at most $n(H + 1) + o(n)$ bits.

adaptive coding

FGK, Vitter's and Gagie's algorithms all use $O(n(H+1))$ total time to encode and decode, however.

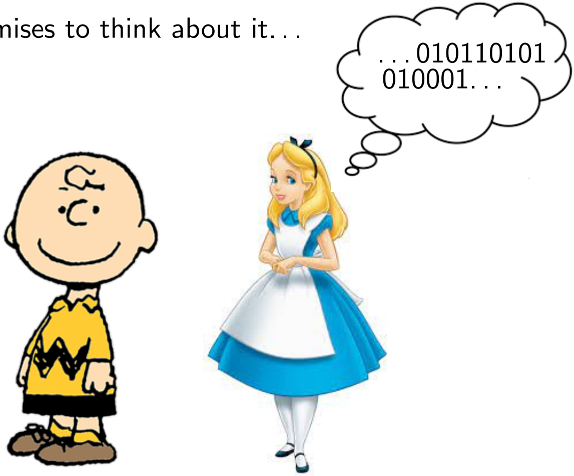
Charlie asks if there isn't an algorithm for adaptive prefix-free coding that uses at most $n(H+1)+o(n)$ bits and $O(n)$ time to encode and decode.

Alice admits there is — Gagie and Nekrich's — but it's never been implemented and it uses a fusion-tree node, so it'll cost him extra.

adaptive coding

Charlie asks if there isn't some simple alternative to Gagie and Nekrich's algorithm with the same bounds (ignoring lower-order terms and assuming $n \gg \sigma$).

Alice promises to think about it...



new algorithm: encoding

To encode a string $S[1..n]$ over an alphabet of size σ:

1. break S into blocks of length $\lceil \sigma \lg n \rceil$
2. encode the first block with an agreed code that assigns a codeword of length $\lceil \lg \sigma \rceil$ to each character
3. for $i > 1$
    1. after encoding the first $i - 1$ blocks, build a $\lceil \lg \sigma + \lg \lg n \rceil$-restricted Shannon code for the distribution of characters in them
    2. build an $O(\sigma)$-space lookup table for encoding
    3. encode the ith block with that table

new algorithm: decoding

To decode S:

1. decode the first block with the agreed code that assigns a codeword of length $\lceil \lg \sigma \rceil$ to each character
2. for $i > 1$
    1. after decoding the first $i - 1$ blocks, build a $\lceil \lg \sigma + \lg \lg n \rceil$-restricted Shannon code for the distribution of characters in them
    2. build an $O(\sigma \log n)$-space lookup table for decoding
    3. decode the ith block with that table

new algorithm: analysis

Building the code for each block takes $O(\sigma)$ time.

Building the table to encode each block takes $O(\sigma)$ time.

Building the table to decode each block takes $O(\sigma \log n)$ time.

Once we have the tables, encoding and decoding each block takes $O(\sigma \log n)$ time.

Since blocks are $\lceil \sigma \log n \rceil$ characters long, we encode and decode S using $O(n)$ total time.

In only 5 minutes, I can't prove the bound on the encoding length or that it's worst-case optimal.

comparison

	authors	encoding length	encoding time	decoding time
unordered	Faller-Gallager-Knuth	$H + 2 + \delta$	$H + 1$	$H + 1$
	Vitter	$H + 1 + \delta$	$H + 1$	$H + 1$
	Gagie	$H + 1$	$H + 1$	$H + 1$
	Karpinski-Nekrich	$H + 1$	1	$\lg H + 1$
	Gagie-Nekrich	$H + 1$	1	1
	new	$H + 1$	1	1
alphabetic	Gagie	$H + 2$	$H + 1$	$H + 1$
	Gagie-Nekrich	$H + 2$	1	$\log \log n$
	Golin et al.	$H + O(1)$	1	?
	new	$H + 2$	1	1

(These bounds are amortized and per character, and don't include $o(1)$ terms in the first column or asymptotic notation in the second and third.)