NAME

Static coders for streams of integers.

Simple: *Simple coding using 64-bit word buffer*

Carry: *Carry coding using 64-bit word buffer*

Slide: *Slide coding using 64-bit word buffer*

PForDelta: *PForDelta coding using 64-bit word buffer*

Byte: *byte-coding*

For each of these programs (except **Byte**) a 32-bit buffer version can be built by changing the Makefile flag **-D_BYTES_PER_WORD=8** to **-D_BYTES_PER_WORD=4**.

In addition, except for **Slide**, a slower no-bulk-decoding version can be built by removing the compiler flag **-DFAST_DECODE** from the Makefile.

For testing, run the script **test-coder** which also demonstrates usage.

SYNOPSIS

<coder> [**−h] [−d] [−t] [−v] [−i** *input_file*] [**−o** *output_file*]

<coder> −x [**−d**] [**−t**] [**−v**] [**−i** *input_file*] [**−o** *output_file*]

DESCRIPTION

The various programs encode and decode binary files of unsigned integers (or text files containing positive integers) using five different static codes. Good compression is achieved if the integers are generally small. Most of the methods are also sensitive to localized clustering.

Input is assumed to be a sequence of independent values. The **−d** option indicates that the input values are monotonically increasing, and in this case consecutive differences are calculated before compression is undertaken.

The main virtue of these implementations is speed − they all obtain plausible compression rates at very high throughput speeds. Output is to *stdout* unless changed with the **−o** option; input is from *stdin* unless changed with the **−i** option.

USAGE

To use **coder** to encode a file *numbers* containing unsigned integers into a file *numbers-enc* and then decode that file to create *numbers-dec* the following two commands are used:

coder < *numbers* > *numbers-enc*

coder −x < *numbers-enc* > *numbers-dec*

The files *numbers* and *numbers-dec* should be the same, and can be checked with

cmp *numbers numbers-dec*

Other options can be seen using **coder −h**

OPTIONS

−h display a help message

−x decode (default is to encode)

−v generate details of the input and output files (encoding only)

−t input (or output if **−x** is specified) is a text file. NB, in this case the output file may differ from the input file because of leading zeros and/or whitespace considerations.

−d input (or output if **−x** is specified) is a monotonically increasing list of integers; what is coded is the differences between consecutive values.

ORIGINS

Simple, **Carry**, and **Slide** are primarily based on research work of the two authors [publication details will be provide here after paper review is complete]. For details of the implementation of these three methods, see the paper; the 32-bit versions of the compression schemes were described elsewhere, see the reference list of our paper.

The **PForDelta** implementation is based on the work of Zukowski, Heman, New, and Boncz, "Super-scalar RAM-CPU cache compression", *International Conference on Data Engineering*, page 59, April 2006; and of Zhang, Long, and Suel, "Performance of compressed inverted list caching in search engines", *Conference on the World Wide Web*, page 387, 2008.

The **Byte** coder has a long history, see, for example, Scholer, Williams, Yiannis, and Zobel, "Compression of inverted indexes for fast query evaluation", *SIGIR International Conference on Research and Development in Information Retrieval*, page 222, 2002.

For general information about compression and coding, see *Compression and Coding Algorithms* A. Moffat and A. Turpin, Kluwer Academic Press, February 2002. Further information about this book is available at http://www.cs.mu.oz.au/caca/

We ask that, if you use this software to derive experimental results that are reported in any way, you cite the original work in which the underlying processes are described (by referencing the listed paper); and also acknowledge our authorship of the implementation you have used.

BUGS

These coders have not been extensively tested, and should be used for research purposes only. Portability of the software or of the compressed files is not guaranteed. There is no warranty, either express or implied, that the software is fit for any purpose whatsoever, and neither the authors nor The University of Melbourne accept any responsibility for any consequences that may arise from your use of this software.

LICENSE

AUTHORS

Vo Ngoc Anh and Alistair Moffat, Department of Computer Science and Software Engineering, The University of Melbourne, Victoria 3010, Australia. Email: vo@csse.unimelb.edu.au, alistair@csse.unimelb.edu.au.