## NAME

rbuc − static coder for streams of integers

## SYNOPSIS

**rbuc** [ **−h** ] [ **−t** ] [ **−d** ] [ **−t** ] [ **−v** ] [ **−o** *file* ]

**rbuc −x** [ **−t** ] [ **−d** ] [ **−t** ] [ **−v** ] [ **−o** *file* ]

## DESCRIPTION

**rbuc** encodes and decodes a file of strictly positive unsigned integers or strictly positive text integers using a static code. Good compression is achieved if the integers are generally small. It is 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 **rbuc** calculates consecutive differences before undertaking the compression.

The main virtue of **rbuc** is speed − it obtains 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 encode a file *numbers* containing strictly positive unsigned integers into a file *numbers-enc* and then decode that file to create *numbers-dec* you would proceed as

**rbuc** < *numbers* > *numbers-enc*

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

The files *numbers* and *numbers-dec* should be the same. (Check with **cmp** *numbers numbers-dec* )

Other options, including a self-evaluation one, can be seen using **rbuc ?**

## 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

**rbuc** is based upon original work of Alistair Moffat and Vo Ngoc Anh, described in "Binary Codes for Non-Uniform Sources", presented at the *2005 IEEE Data Compression Conference*, Snowbird, March 2005, pages 133-142. See http://www.cs.mu.oz.au/˜alistair/abstracts/ for additional publication information. Note, however, that in the presentation given in that paper the integers being coded were assumed to be non-negative; in the software implementation all values are first shifted by one, and the input stream must be strictly positive values (that is, one or greater, no zeroes).

For details of the implementation, see the paper listed above. 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

**rbuc** has not been extensively tested, and should be used for research purposes only. Portability is not guaranteed. There is no warranty, either express or implied, that it 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@cs.mu.oz.au, alistair@cs.mu.oz.au.