Advanced Data Management for Data Analysis

Assignment 1

22.10.2019

Due: *Sunday, 10 November 2019, 23:59 CET*
Notes:

- Work in groups of max. 3 students.
- Produce a PDF document describing the functionality, implementation and use of your program.
- Accompany the document with a compressed archive containing your programs entire source code as well as any accompanying scripts (if any).
- Name your submission files
  ADM2019_A1_<student_id_1>_<student_id_2>_<...>_<student_id_N>_report.pdf,
  ADM2010_A1_<student_id_1>_<student_id_2>_<...>_<student_id_N>_archive.zip
  (with student IDs sorted in ascending order!)
- Submit both files by email
  - To: S.Manegold@liacs.leidenuniv.nl
  - Subject: [ADM-2019] Assignment 1 (<list of student IDs *in ascending order*>)

Points:

This assignment is worth a total of 100 points, 10 points for the encoding and decoding for each of the five encoding schemes (equally split between the actual source code and it documentation, i.e., the report), plus max. 10 bonus points (see details below). The final score (grade) will be point devided by 10 to fit in the 0-10 grade system.

Please read the entire assignment instructions carefully and make sure you follow them properly.

Your task is to write a stand-alone program that applies some of the data compression techniques we discussed in class to given data sets. While C or C++ are preferred / recommended, you can also use an other programming language of your choice, provided I can compile and run your program on a standard Fedora 30 Linux distribution.

The data sets are provided as single-column CSV (comma-separated values) files using line-end as record separator, i.e., text files that contain one value per row. Please be aware that the files are originally created on a Linux system, i.e., merely a single line-feed character (LF; ASCII: 10 (dec) / 0A (hex)) is used as end-of-line character, rather than Windøws-typical carriage-return plus line-feed (CRLF; ASCII 13,10 (dec) / 0d,0a (hex)). The data type (se below) is encoded in the file name.
For space and bandwidth efficiency, all data files are provided in zip-compressed format. Please decompress them locally before using them with your program.

Your program should accept three mandatory command line arguments:

1. "**en**" or "**de**" to specify whether your program should **en**code the given data or **de**code data that your has encoded; with **en**coding, all five encodings (see below) should be produced, with **de**coding, the type of encoding can be derived from the file name suffix (see below);
2. the **name (or entire path) of the file** to be en- or de-coded;
3. the data type of the input data:
   - "**int8**": 8-bit (1-byte) integer,
   - "**int16**": 16-bit (2-byte) integer,

- “**int32**”: 32-bit (4-byte) integer,
- “**int64**”: 64-bit (8-byte) integer,
- “**string**”: character string of arbitrary length.

For encoding, your program should read the data from the given text (CSV) file from disk, and write the data back to disk in the following encoding formats (if applicable for the given data type); the output file name should be the input file name suffixed with the encoding acronym:

- “.**bin**”: uncompressed binary format (integer types, only),
- “.**rle**”: run-length encoding,
- “.**dic**”: dictionary encoding,
- “.**for**”: frame of reference encoding (integer types, only),
- “.**dif**”: differential encoding (integer types, only).

For decoding, your program should read the encoded data from the given encoded file, and output the plain decoded data (text) to the console (stdout).

For each encoding, it is sufficient to use standard byte-aligned data types, i.e., code widths of 1/2/4/8 byte (8/16/32/64 bit). Improving the compression ratio by using sub-byte aligned/wide codes and “bit packing” would yield bonus points (one for encoding and decoding per encoding scheme, i.e., max. ten in total), if done correctly.

You need to accompany your program with a report that documents how your code works, i.e., how you implemented each of the encoding techniques, how your program needs to be compiled, and how to run your program.
Additionally, your report should list the sizes of all input files and the respective output files for all applicable encoding techniques, and compare and discuss the different compression ratios.