tcmpr Documentation

Release 0.1.post0.dev8+gc8e1adc

Konrad Poreba

Jun 26, 2019

Contents

This is the documentation of **tcmpr**.

Program compress and decompress text files, using chosen algorithm. It can run on files as well as in recursive way on directory with text files. Implemented algorithms are:

- Huffmann (Huffman coding)
- LZSS (Lempel–Ziv–Storer–Szymanski)
- LZW (Lempel–Ziv–Welch)
- Shannon (Shannon coding)

CHAPTER 1

Contents

1.1 Installation

1.1.1 Requirements

The installation of tcmpr only requires a recent version of pip and python 3.

Note: Program is distributed as binary wheel package, so all dependencies are included.

1.1.2 Installation

Make sure you have `pip` installed, then simply type:

```
$ pip install tcmpr
```

The most recent version can be installed with:

```
$ pip install --upgrade tcmpr
```

Using `pip` also has the advantage that all requirements are automatically installed.

1.1.3 Uninstall

If you would like to uninstall tcmpr use following command:

```
$ pip uninstall tcmpr
```

1.2 Getting Started

tcmpr is a text compression program that uses most popular lossless algorithms. It can compress using following algorithms:

- Huffman coding
- LZSS
- LZW
- Shannon coding

Note: It's not the fastest program to compression, rather should be considered as demonstration of basic lossless algorithms.

1.2.1 Quick start

To install *tcmpr* use following command (assuming that you have *pip* installed):

```
$ pip install tcmpr
```

Compress file *filename.txt* with default algorithm (huffman coding):

```
$ tcmpr filename.txt
```

Decompress file *filename.txt.huffman* with appropriate algorithm based on extension (here *.huffman*):

```
$ tcmpr -d filename.txt.huffman
```

1.2.2 Parameters

Compress file *filename.txt* with chosen algorithm (here with huffman coding):

```
$ tcmpr -alg=huffman filename.txt

or

$ tcmpr --algorithm=huffman filename.txt
```

1.2.3 Help

Help can be invoked in following way:

1.3 License

1.4 Contributors

- Konrad Poreba <konrad@porebasoftware.pl>

1.5 Changelog

1.5.1 Version 0.1

- First release

1.6 tcmpr

1.6.1 tcmpr package

Subpackages

tcmpr.algorithms package

Subpackages

tcmpr.algorithms.entity package

Submodules

tcmpr.algorithms.entity.node module

Module contain Node entity used in binary tree which are applied in compression algorithms

class tcmpr.algorithms.entity.node.**Node**(*char*, *frequency*, *left=None*, *right=None*)
Bases: object

Class representing a Node data structure in the binary tree

is_leaf()

Module contents

tcmpr.algorithms.huffman package

Submodules

tcmpr.algorithms.huffman.compressor module

This module is responsible for implementing HUFFMAN CODING algorithm which can be used to compressing input file.

tcmpr.algorithms.huffman.compressor.**build_binary_tree**(*frequencies*)
Function to create binary huffman tree based on counted frequencies in input data

tcmpr.algorithms.huffman.compressor.**compress_huffman**(*input_file*)
Function compress input file and create new compressed file in the same folder (or in defined output path). Algorithm uses a priority queue where the node with lowest probability (frequency) is given highest priority. :param: input_file, :return: compressed output file path

tcmpr.algorithms.huffman.compressor.**put_frequencies_into_priority_queue**(*frequencies*)
Create PriorityQueue and put there frquencies

tcmpr.algorithms.huffman.decompressor module

tcmpr.algorithms.huffman.decompressor.**decompress_huffman**(*input_file*)

Module contents

tcmpr.algorithms.lzss package

Submodules

tcmpr.algorithms.lzss.compressor module

This module is responsible for implementing LZSS CODING algorithm which can be used for compression of input file

tcmpr.algorithms.lzss.compressor.**compress_lzss**(*input_file*)

Module contents

tcmpr.algorithms.lzw package

Submodules

tcmpr.algorithms.lzw.compressor module

This module is responsible for implementing LZW CODING algorithm which can be used to compressing input file.

tcmpr.algorithms.lzw.compressor.**compress_lzw**(*input_file*)
Function compress input file and create new compressed file with extension ".lzw" (for decompression purposes) in place of use. Algorithm create dictionary of encoding base on input file and frequent subsequence of chars. :param input_file: :return: compressed file path

tcmpr.algorithms.lzw.compressor.**encode_data**(*data*)
Create dictionary base on input data and encode data and look for frequent subsequences and assign them to proper codes in dictionary. :param data: :return:

tcmpr.algorithms.lzw.decompressor module

This module is responsible for implementing LZW CODING algorithm which can be used to decompress input file with extension ".lzw".

class tcmpr.algorithms.lzw.decompressor.**Step**
Bases: object

get_step()

set_step(*step*)

tcmpr.algorithms.lzw.decompressor.**decode_codes**(*codes*)

tcmpr.algorithms.lzw.decompressor.**decompress_lzw**(*input_file*)
Function to decompress input file with extension ".lzw"

Module contents

tcmpr.algorithms.utils package

Submodules

tcmpr.algorithms.utils.tools module

tcmpr.algorithms.utils.tools.**convert_bytes_to_bit_str**(*byte_array*)
Convert bytes to bits string for decoding purpose

tcmpr.algorithms.utils.tools.**get_byte_array**(*padded_encoded_string*)
Convert padded encoded string into bytes

tcmpr.algorithms.utils.tools.**get_codes_from_binary_tree**(*root*)
Return code for each char in input data (original)

tcmpr.algorithms.utils.tools.**get_decoded_str**(*root*, *encoded_string*)
Decode encoded string and return it

tcmpr.algorithms.utils.tools.**get_encoded_str**(*root*, *data*)
Change data to encoded string and return it

tcmpr.algorithms.utils.tools.**pad_encoded_str**(*encoded_string*)
Return padded encoded string with zero to full byte if missing

tcmpr.algorithms.utils.tools.**remove_padding_of_encoded_str**(*padded_encoded_string*)
Remove padding to get original encoded string

Module contents

Module contents

Submodules

tcmpr.application module

Main module for tcmpr application used to run program and parse arguments.

tcmpr.application.**compress**(*file_name*, *algorithm*)
Pick proper algorithm chosen by the user and compress input file

tcmpr.application.**decompress**(*file_name*)
Pick proper algorithm to decompress file based on coding extension attached to compressed file like:

.huffman .lzw .lzss .shannon

tcmpr.application.**main**(*args*)
Main entry point allowing external calls

Parameters **args** ([*str*]) – command line parameter list

tcmpr.application.**run**()
Entry point for console_scripts

tcmpr.application.**setup_logging**(*loglevel*)
Setup basic logging

Parameters **loglevel** (*int*) – minimum loglevel for emitting messages

tcmpr.argument_parser module

tcmpr.argument_parser.**create_parser**()
Function that create parser, helpful in writing test cases

tcmpr.argument_parser.**parse_args**(*args*)
Parse command line parameters

Parameters **args** ([*str*]) – command line parameters as list of strings

Returns command line parameters namespace

Return type argparse.Namespace

Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index