
tcmpr Documentation

Release 0.1.post0.dev8+gc8e1adc

Konrad Poreba

Jun 26, 2019

Contents

1	Contents	3
1.1	Installation	3
1.2	Getting Started	4
1.3	License	4
1.4	Contributors	5
1.5	Changelog	5
1.6	tempr	5
2	Indices and tables	9
	Python Module Index	11
	Index	13

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](#))

1.1 Installation

1.1.1 Requirements

The installation of tcpr 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 tcpr
```

The most recent version can be installed with:

```
$ pip install --upgrade tcpr
```

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

1.1.3 Uninstall

If you would like to uninstall tcpr use following command:

```
$ pip uninstall tcpr
```

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

The MIT License (MIT)

Copyright (c) 2019 Konrad Poreba

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

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 frequencies

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`

t

- `tcmpr`, 8
- `tcmpr.algorithms`, 8
- `tcmpr.algorithms.entity`, 6
- `tcmpr.algorithms.entity.node`, 5
- `tcmpr.algorithms.huffman`, 6
- `tcmpr.algorithms.huffman.compressor`, 6
- `tcmpr.algorithms.huffman.decompressor`, 6
- `tcmpr.algorithms.lzss`, 6
- `tcmpr.algorithms.lzss.compressor`, 6
- `tcmpr.algorithms.lzw`, 7
- `tcmpr.algorithms.lzw.compressor`, 6
- `tcmpr.algorithms.lzw.decompressor`, 7
- `tcmpr.algorithms.utils`, 8
- `tcmpr.algorithms.utils.tools`, 7
- `tcmpr.application`, 8
- `tcmpr.argument_parser`, 8

B

`build_binary_tree()` (in module *tcmpr.algorithms.huffman.compressor*), 6

C

`compress()` (in module *tcmpr.application*), 8

`compress_huffman()` (in module *tcmpr.algorithms.huffman.compressor*), 6

`compress_lzss()` (in module *tcmpr.algorithms.lzss.compressor*), 6

`compress_lzw()` (in module *tcmpr.algorithms.lzw.compressor*), 6

`convert_bytes_to_bit_str()` (in module *tcmpr.algorithms.utils.tools*), 7

`create_parser()` (in module *tcmpr.argument_parser*), 8

D

`decode_codes()` (in module *tcmpr.algorithms.lzw.decompressor*), 7

`decompress()` (in module *tcmpr.application*), 8

`decompress_huffman()` (in module *tcmpr.algorithms.huffman.decompressor*), 6

`decompress_lzw()` (in module *tcmpr.algorithms.lzw.decompressor*), 7

E

`encode_data()` (in module *tcmpr.algorithms.lzw.compressor*), 7

G

`get_byte_array()` (in module *tcmpr.algorithms.utils.tools*), 7

`get_codes_from_binary_tree()` (in module *tcmpr.algorithms.utils.tools*), 7

`get_decoded_str()` (in module *tcmpr.algorithms.utils.tools*), 7

`get_encoded_str()` (in module *tcmpr.algorithms.utils.tools*), 7

`get_step()` (*tcmpr.algorithms.lzw.decompressor.Step* method), 7

I

`is_leaf()` (*tcmpr.algorithms.entity.node.Node* method), 5

M

`main()` (in module *tcmpr.application*), 8

N

Node (class in *tcmpr.algorithms.entity.node*), 5

P

`pad_encoded_str()` (in module *tcmpr.algorithms.utils.tools*), 7

`parse_args()` (in module *tcmpr.argument_parser*), 8

`put_frequencies_into_priority_queue()` (in module *tcmpr.algorithms.huffman.compressor*), 6

R

`remove_padding_of_encoded_str()` (in module *tcmpr.algorithms.utils.tools*), 7

`run()` (in module *tcmpr.application*), 8

S

`set_step()` (*tcmpr.algorithms.lzw.decompressor.Step* method), 7

`setup_logging()` (in module *tcmpr.application*), 8

Step (class in *tcmpr.algorithms.lzw.decompressor*), 7

T

tcmpr (module), 8

tcmpr.algorithms (module), 8

tcmpr.algorithms.entity (module), 6

tcmpr.algorithms.entity.node (module), 5

tcmpr.algorithms.huffman (*module*), 6
tcmpr.algorithms.huffman.compressor
 (*module*), 6
tcmpr.algorithms.huffman.decompressor
 (*module*), 6
tcmpr.algorithms.lzss (*module*), 6
tcmpr.algorithms.lzss.compressor (*mod-
ule*), 6
tcmpr.algorithms.lzw (*module*), 7
tcmpr.algorithms.lzw.compressor (*module*),
 6
tcmpr.algorithms.lzw.decompressor (*mod-
ule*), 7
tcmpr.algorithms.utils (*module*), 8
tcmpr.algorithms.utils.tools (*module*), 7
tcmpr.application (*module*), 8
tcmpr.argument_parser (*module*), 8