Digital Communication 2018-19
Lempel-Ziv Compression

Documentation

There are two source files come with this documentation: LZ77.py and simulator.py. The first file defines the class of LZ77 compression algorithm that integrated with compress(input_file_path, output_file_path) and decompress(input_file_path, output_file_path) functions, and the other imports LZ77.py and provides you an interface to run the simulation.

LZ77.py
A simplified implementation of the LZ77 Compression & Decompression Algorithm. My implementation of LZ77 algorithm is able to compress or decompress any file type.

compress(input_file_path, output_file_path)
Given the path of an input file, its content is compressed by applying a simple LZ77 compression algorithm. Before compression, two parameters are attached to the output_buffer: the window_size in binary representation and the lookahead_buffer_size in binary representation. Both parameters are stored in 10 bits (from 0 to 1023). During compression, the compressed format is 1 bit flag, followed by 8 bits (1 byte) character when there are no previous matches within window. Otherwise, the algorithm will attach 1 bit flag, followed by {param: window_size_bit} bits for distance to the start of the match from the current position, {param: lookahead_buffer_size} bits for length of the match, and 8 bits (1 byte) character. Finally, the compressed data is written into a binary file for which the path is provided.

decompress(input_file_path, output_file_path)
Given the path of a compressed file, its content is decompressed by applying a simple LZ77 decompression algorithm. Before decompression, the first 20 bits of data are read for decompression purpose: the window_size in binary representation (the first 10 bits) and the lookahead_buffer_size in binary representation (the other 10 bits). During decompression, the decoder reads the flag to check whether there is a match. If there is evidence of match, the algorithm will read the next 8 bits character and append it to the output_buffer. Otherwise, the algorithm will read the next {param: window_size_bit} bits for distance to the start of the match from the current position, followed by {param: lookahead_buffer_size} bits for length of the match, and 8 bits (1 byte) character. The algorithm will append the match and character at the end of output_buffer. Finally, the decompressed data is written into a binary file for which the path is provided.

simulator.py
This file provides you an interface to test the LZ77.py implementation. You can import the package by adding `from LZ77 import LZ77Compressor` to the head. After that you can create your own compress object: `compressor = LZ77Compressor(W, L)`. Please note that W should be in the range from 0 to 1023 and L should be in the range from 15 to 1023. Finally, you can start compress or decompress your file by `compressor.compress(input, output)` or `compressor.decompress(input, output)` commands.