Implementation of LZW algorithm by using C, C++, python or other program

<https://codecogs.com/library/computing/io/compression/lzw.php>

**Overview**

This module allows compressing or decompressing an input array using the Lempel-Ziv-Welch (LZW) algorithm. The LZW algorithm is a lossless data compression algorithm created by Terry Welch in 1984. This algorithm represents an improved version of the LZ78 algorithm created by Abraham Lempel and Jacob Ziv in 1978.

The idea of the compression algorithm is the following: as the input data is being processed, a dictionary keeps a correspondence between the longest encountered words and a list of code values. The words are replaced by their corresponding codes and so the input file is compressed. Therefore, the efficiency of the algorithm increases as the number of long, repetitive words in the input data increases.

**Note:** Either when using the compression or the decompression methods, the elements of the input array must be of type **unsigned** **char**, which is also the type of the resulting array's elements.

The following example generates a sample array of N random letters (from A to Z) and compresses it. The compressed array is then decompressed to see if the sample array is identical to the uncompressed array. The size of the compressed array is also displayed, to prove the efficiency of the LZW algorithm.

**Example 1**

**#include <[codecogs/computing/io/compression/lzw.h](https://codecogs.com/pages/pagegen.php?id=913" \t "_blank)>**

**#include <iostream>**

// the number of characters to generate in the sample array

**#define N 10000**

**using** **namespace** Computing::IO::Compression;

**int** main()

{

  // initialize random seed

  srand(time(0));

  // generate an array of N random letters

  std::vector<**unsigned** **char**> sample;

**for** (**int** i = 0; i < N; ++i)

    sample.push\_back('A' + rand() % ('Z' - 'A' + 1));

  // compress the sample array

  std::vector<**unsigned** **char**> compressed = LZW::compress(sample);

  // decompress the compressed array

  std::vector<**unsigned** **char**> uncompressed = LZW::decompress(compressed);

  // compare the sizes of the compressed and uncompressed arrays

  std::cout << "      Size of the sample array: " << N << std::endl;

  std::cout << "  Size of the compressed array: " << compressed.size() << std::endl;

  std::cout << "Size of the uncompressed array: " << uncompressed.size() << std::endl;

  std::cout << std::endl;

  // test if the sample and the uncompressed arrays are identical

  // this proves that the LZW compression algorithm does not affect the initial data

**bool** identical = (N == uncompressed.size());

**for** (**size\_t** i = 0; identical && i < uncompressed.size(); ++i)

**if** (sample[i] != uncompressed[i])

      identical = **false**;

**if** (identical)

    std::cout << "The sample and uncompressed arrays are identical." << std::endl;

**else**

    std::cout << "Error! The sample and uncompressed arrays are NOT identical." << std::endl;

**return** 0;

}