- **Lossless Data Compression**

In this type, data will encode at the sender point and then relatively it will reduce its size in large proportions. Then retrieve it at the recipient point without loss of information. So, after decompress the file, the zip file must be completely identical to the original file, hence the name of data keeper compression. This kind of compression is used in execution files EXE, and text files DOC and TXT and others.

There are two ways to encode and retrieve data:

1. **Dictionary method**

In this way it will built dictionary from words and letters that make up the message where the message is encoded by a number representing its position in the dictionary, so the message sent representing words locations numbers in the dictionary. For data compression needs, the most common vocabulary are placed at the beginning of the dictionary. This requires prior knowledge of the probability of the vocabulary occurrence, and the existence of the dictionary before the transmitter with the sender and the recipient. This method is not much use. *Lempel–Ziv–Welch (LZW) is a universal lossless data compression algorithm created by Abraham Lempel, Jacob Ziv, and Terry Welch.* (LZW) Algorithm is considered one of the algorithms in this method, which is used a dictionary with an entrance of 4096 capacity. The first 256 of an entrance is used for the codes. On the recipient side, it will receipt the dictionary at first, then it will receipt a numbers which represents a message codes, that would be translated and re-composition.

2. **Statistical method**

It is relying on the recurrence of the letters sent in encoded of data, and there are two ways:

i. Arithmetic Coding:

It is compute the probability of appearance of each letter of the message and within a certain range. This method is based on the principle of replacing the message by number of decimal according to the algorithm used.

ii. Iterative way RLE:

Run Length Encoding (RLE) is a very simple lossless data compression algorithm. The idea behind this method is to encode long sequences of the same data values with the shortest possible encoding. If a character is repeated a number of times (repeating values is called a Run), the sequence can be replaced

with information about the number of occurrences of the same consecutive character (length of the run).

While decompression back to the original, the string is reconstructed by repeating the character for the number of its occurrence noted in compressed file.

REL Procedure :

- If the same character occurred at least four times then count the number of occurrences.
- Write compressed data in the format : "counted character ! number of its occurrences" where ! is a special character used to denote that he number following it is the number of occurrences.
- Construct the compressed data sequence.

***Example*:** Assume an uncompressed sequence ABCCCCCCCCDEFFFFFFGGG (21 byte). In this case character C is repeated more than 4 times. Thus C is replaced with C!8.

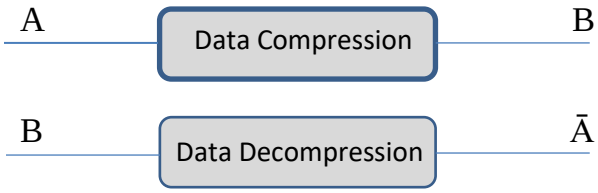
Thus the compressed string is become ABC!8DEF!6GGG (13 byte).

- **Lossy Data Compression**

This technique is used when there is a desire to get a very high compression ratio and there is no urgent need for the output's file after the compression process to be completely identical to the original file. This means that the compressed file, after decompressing will not be similar to the original file, but are getting of 90% or 80% (Only the important information from the original file), that is to say the quality of the file will be less than the quality of the original file. This kind of compression suitable to multimedia files, such as audio, image and video files because such files are characterized by large size and lack of brevity, that is, at a time which there is a very large amount of data will be a lot of extra data that deleted it does not affect in the informational level of the data sent, that is why the knowledge and delete it will not affect too much.

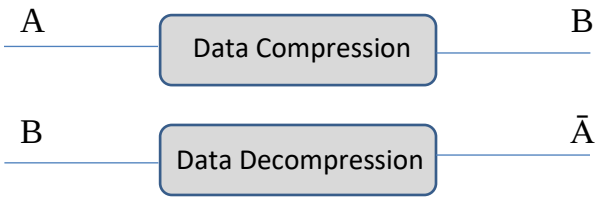
There are several efficient algorithms in this kind of compression, including the JPE for image compression and Real Media to compress video files and MP3 to compress audio files.

Lossless Data Compression



Ā = A

Lossy Data Compression



$\bar{A} \approx 90\% A$

***Example*:** You have the following string: 88888444. Express by using the first method and the second method to rewrite the string in less format.

1. Lossy Data Compression …..84 Less size but data loss
2. Lossless Data Compression…..8(5)4(3) Less size but there is no data Loss

Coding theory: Coding theory means replacing the letters and symbols by using a binary code (0,1). It is a necessary process for the purpose of efficient transmitter, and also because of the presence of digital transmitters. The encoding process seeks to achieve the following goal:

1. High-speed connection (Source coding methods).
2. The fewest mistakes and the ability to detect and correct errors (Channel Coding methods).

Information theory has been able to achieve the two goals in a balanced manner by following the methods of encoding based mathematical equations. To achieve the first goal is to follow the ways of the **Source Coding Method**, and to achieve the second goal is to follow the ways of the **Channel Coding Method**.

o Desired Properties in Coding

You must take into account some attributes that support the encoding process. The desired attributes in the encoding process are:

1. ***Singularity:*** The encoding process should give one group or series of binary codes for each symbol (letter) of the language symbols.

***Example*:** Suppose you have a language consisting of three letters a, b, and c.

- The Encoding **a→0 , b→1, and c→0** is not true for lack of the uniqueness property because both of a and c have the same code word.
- The Encoding **a→0 , b→1, and c→01** is true because it have the uniqueness property.

2. ***Non – Ambiguity:*** The Encoding Process should be unambiguous, in other words, the symbols should not mix with when decoding and re-formation words and sentences.

***Example*:** Suppose you have a language consisting of three letters a, b, and c.

- The Encoding **a→1 , b→00, and c→11** is not true because it have ambiguity. When decoder "1111", there will be two possibilities to distinguish letter. The first possibility is that the message is "cc". The second possibility is that the message will be "aaaa". So, this is the reason for the ambiguity, because both of which have the same code word.
- The Encoding **a→0 , b→01, and c→011** is true, because they have both of two properties, uniqueness and non-ambiguity properties.

3. ***Instantaneous:*** Decoding process must be immediately in order to re-form words and sentences. Where directly retrieve all the letters after full receipt of the code word.

***Example*:** Suppose you have a language consisting of four letters a, b, c, and d.

- The Encoding **a→0 , b→01, c→011, and d→0111** is not Instantaneous, because of that , after reading the string "0" can't figure out if it's "A" character encoding or whether it is a prefix for each of the characters "B" and "C".
- The Encoding a→0 , b→10, c→110, and d→111 is true, because they have both of uniqueness, non-ambiguity, and Instantaneous properties. Meaning it is an ideal encoding, in which all the desired attributes are available.

o Coding Efficient

To determine the efficiency is depending on the coding efficiency, which is calculated from the following equation:

$$E = H / L$$

Where H : Entropy of the language.

L : Coding rate, which is calculated as follows:

$$L = \sum_{i=1}^{n} P_i * L_i$$

Where Li : Represents the number of Bits used to represent the character i.

Note:

$$E \propto \frac{1}{L}$$