**6. Compression:**

6.1. Reasons To Use Data Compression Algorithms:

we need Data Compression mainly because:

* Uncompressed data can take up a lot of space, which is not good for limited hard drive space and internet download speeds.
* While hardware gets better and cheaper, algorithms to reduce data size also helps technology evolve.

there are many techniques for data compressing, but in this project we will use ZLW technique to compress & decompress Xml files.

6.2. Lempel–Ziv–Welch (LZW) Algorithm:

LZW algorithm is a very common compression technique. This algorithm is typically used in GIF and optionally in PDF and TIFF. Unix’s ‘compress’ command, among other uses. It is lossless type of compression, meaning no data is lost when compressing. The algorithm is simple to implement and has the potential for very high throughput in hardware implementations.

The Idea relies on reoccurring patterns to save data space. LZW is the foremost technique for general purpose data compression due to its simplicity and versatility.

6.3. How does it (LZW) Algorithm work?

LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. Because the codes take up less space than the strings they replace, we get compression. Characteristic features of LZW includes,

* When encoding begins the code table is empty. Compression is achieved by using codes 0 (can be started by 256 entries) through 4095 to represent sequences of bytes.
* As the encoding continues, LZW identifies repeated sequences in the data, and adds them to the code table.
* Decoding is achieved by taking each code from the compressed file and translating it through the code table to find what character or characters it represents.

6.4. Implementation:

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.

6.4.1. LZW ENCODING:

PSEUDOCODE:

1 Initialize an empty table (list of strings, each string has its numeric counterpart which is the index where this string is inserted in the list.

2 P = first input character

3 WHILE not end of input stream

4 C = next input character //if P is not in the string table add it in the table.

5 IF P + C is in the string table

6 P = P + C

7 ELSE

8 output the code for P

9 add P + C to the string table

10 P = C

11 END WHILE

12 output code for P

6.4.2. Decompression

In our code, the decompressing code is different from the decompression method in the LZU method, as we reuse the table we filled in in the encoder function and the codes (numbers) corresponding to each encrypted segment within the file, and use them backwards through the table, where each encrypted code is replaced with the corresponding string in the string table.

Reference: [LZW (Lempel–Ziv–Welch) Compression technique - GeeksforGeeks](https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/)