0.1 Lempel Ziv Welch

- *Algorithm:* LZW Encoding (algo. 1), LZW Decoding (algo. 2)
- *Input:* A string for encoding, or a list of integers ranging from 0–4095 (ASCII representation) for decoding
- *Complexity:* $\mathcal{O}(n)$
- *Data structure compatibility:* N/A
- *Common applications:* Unix file compression and used in GIF image format
- *Note:* LZW stands for "Lempel Ziv Welch"

Problem. Lempel Ziv Welch

Encoding: Given a string s, compress it into a list of ASCII code.

Decoding: Given a list of ASCII code, recover the original string.

Description

Nowadays, almost everyone needs compression, the reasons for this are as follows:

- Original data without compression may occupy a lot space on disk, limiting the space to put important things. Also, without compression, if we want to download some files from the internet, it will take quite a long time since network speed is limited.
- Algorithms of reducing data size like compressing can help improve the development of hardwares.

In this sense, the Lempel-Ziv-Welch Algorithm is introduced as a lossless compression algorithm for reducing data size. In practice, an English text file with large size can be compressed using LZW and its size can be reduced to about half the original size.

Method of LZW compression/Encoding

LZW compression works as reading input of string containing different symbols, and then converting the strings of symbols into codes, usually ranging from 0 to 4095 [2]. To be more specific, the following steps show the exact procedure of LZW compression:

- First, create a code table, with 4096 the usual number of entries. Codes 0 – 255 represents the ASCII codes for all the symbols in character.
- Then read characters of the input string one by one. If the sequence of the previous character and the current one does exist in the table for corresponding code, then assign a new entry in the table of this sequence, and keep the sequence to the next iteration. If the sequence does not have a corresponding entry in the table, just output the code for the current sequence, and assign the sequence as the current character, and resume the remaining iterations [1].
- Finally, after reading the last character of the string, output the last sequence's code.

Method of LZW Decompression/decoding

The steps for LZW decompression is shown below:

- Create a default table containing indexes ranging from 0 – 255 and their corresponding characters.
- Then it operates similarly as the compression method, just reverse the code into original string.

Time Complexity

Since we just need the number of iterations equal to the length of input string, the time complexity for both compression and decompression is $\mathcal{O}(n)$.

Applications

The Graphics Interchange Format (GIF) uses LZW for encoding and decoding. Different codes in the color stand for different pixels, and all of which would be combined as an animation picture.

Pseudocode for LZW Encoding

Algorithm 1: LZW Encoding

Input : A string of characters s
Output: a list of codes compressed from s

```
table ← a table with 256 single characters as keys and 0 – 255 as values;
output ← an empty list;
c ← s[0];
index ← 256;
for i = 1 to s.length − 1 do
    c′ ← s[i];
    if c + c′ is in table then
        c ← c + c′;
    else
        push c into output;
        table[c + c′] ← index;
        index ← index + 1;
        c ← c′;
    end if
end for
return output;
```

Pseudocode for LZW Decoding

Algorithm 2: LZW Decoding

Input : A list l of codes
Output: The original string s compressed into l

```
table ← a table with 0 – 255s as keys and 256 single character as values;
s ← an empty string;
c ← l[0];
s ← s+ table[c];
index ← 256;
for i = 0 to l.length − 1 do
    c′ ← l[i];
    if c′ not in table then
        temp ← table[c];
        temp ← temp + x;
    else
        temp ← table[c′];
    end if
    s ← s+ temp;
    x ← temp[0];
    table[index] ← table[c] + x;
    index ← index + 1;
    c ← c′;
end for
return s;
```

References.

[1] Welch. "A technique for high-performance data compression". In: *Computer* 17.6 (1984), pp. 8–19. DOI: 10.1109/mc.1984.1659158 (cit. on p. 1).

[2] J. Ziv and A. Lempel. "Compression of individual sequences via variable-rate coding". In: *IEEE Transactions on Information Theory* 24.5 (1978), pp. 530–536. DOI: 10.1109/tit.1978.1055934 (cit. on p. 1).