Lempel-Ziv-Storer-Szymanski (LZSS)

LZSS is a derivative of the LZ77 compression algorithm. Instead of storing tuples, it uses a one-bit flag in front of the data to specify if it is a literal or if it is a reference.

If the first bit is 1, the next 8 bits are a literal.

If the first bit is 0, the next 16 bits represents a reference. The reference is as follows (Offset, length)

LZSS does not allow you to extend pass the original size of the copy buffer when you are copying. This is different from LZ77 and in turn, LZSS suffers in some cases due to that.

Let’s use the string ABAABACBCCBABC as an example

|  |  |  |
| --- | --- | --- |
| Buffer | Encoded Data | Final String |
|  | A | 1 01000001 |
| A | B | 1 01000010 |
| AB | A | 1 01000001 |
| ABA | (3, 3) | 0 00000011 00000011 |
| ABAABA | C | 1 01000011 |
| ABAABAC | B | 1 01000010 |
| ABAABACB | C | 1 01000011 |
| ABAABACBC | (3, 2) | 0 00000011 00000010 |
| ABAABACBCCB | (8, 2) | 0 00001000 00000010 |
| ABAABACBCCBAB | C | 1 01000011 |
| ABAABACBCCBABC | N/A | N/A |

Final Encoded data:

10100000 11010000 10101000 00100000 00110000 00111010

00011101 00001010 10000110 00000011 00000010 00000100

00000001 01010000 11000000

(\*Note, Red indicates additional padding)

As Bytes:

0xA0 0xD0 0xA8 0x20 0x30 0x3A 0x1D 0x0A 0x86 0x03 0x02 0x04 0x01 0x80 0xC0

The original output:

0x65 0x66 0x65 0x65 0x66 0x65 0x67 0x66 0x67 0x67 0x66 0x65 0x66 0x67

This algorithm only added one additional byte to the data unlike LZ77 which added 6 additional bytes. LZSS benefits from the same things that LZ77 does in terms of compression. If you were to allow references to go beyond the buffer size like LZ77 does, you would achieve superior compression.