Lempel-Ziv 1977 (LZ77)

LZ77 Compression is sometimes referred to as the sliding window. Instead of storing an explicit dictionary, it instead uses a pointer system to refer to previous point in your data.

In LZ77, you use tuples for your data. Using the pointers, you can also repeat data very easily since every byte you write gets added to the buffer that can be copied. This included data when you are copying.

Let’s use the string ABAABACBCCBABC as an example

|  |  |  |
| --- | --- | --- |
| Buffer | Encoded Data | Final String |
|  | (0, 0, A) | 0x00 0x00 0x65 |
| A | (0, 0, B) | 0x00 0x00 0x66 |
| AB | (2, 1, A) | 0x02 0x01 0x65 |
| ABAA | (3, 2, C) | 0x03 0x02 0x67 |
| ABAABAC | (3, 1, C) | 0x03 0x01 0x67 |
| ABAABACBC | (3, 2, A) | 0x03 0x02 0x65 |
| ABAABACBCCBA | (5, 2, NULL)  OR  (2, 1, C) | 0x05 0x02  OR  0x02 0x01 0x67 |

Final Encoded data:

0x00 0x00 0x65 0x00 0x00 0x66 0x02 0x01 0x65 0x03 0x02 0x67 0x03 0x01 0x67 0x03 0x02 0x65 0x05 0x02

OR

0x00 0x00 0x65 0x00 0x00 0x66 0x02 0x01 0x65 0x03 0x02 0x67 0x03 0x01 0x67 0x03 0x02 0x65 0x02 0x01 0x67

The original output:

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

This algorithm suffers from the fact that you must store at least 3 values even if you only need to store a literal. It is best to not store null data like in the example. Instead, be consistent in your data and use the second case.

It benefits in bit reduction as well. When using bit reduction along with LZ77, you can reduce the additional space requirement when storing literals. Also, if you use a non-greedy method to find the references (meaning that you pick smaller references) there will be more consistency in your data allowing them to also benefit from bit reduction.

When decompressing you copy from the final output and as you copy, you add to the final output. This allows you to compress runs of data very easily.