# Exploring Dictionary-Based Compression: LZ77, LZ78 & LZW

Names / Numbers:

1 - In many cases, the information source produces recurring patterns. How does dictionary-based compression explore this property?

2 – What are the principles behind Tunstall codes?

3 - LZ77 uses a sliding window. What are the roles of the search buffer and lookahead buffer?

4 - What type of dictionary is built in LZ78? How is it different from LZ77?

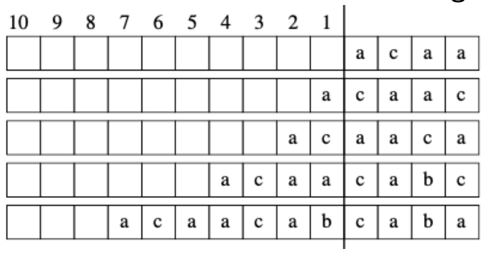
5 - Why does LZW start with a dictionary containing all single-character entries?

6 - What are the main advantages of LZW over LZ78?

7 - Consider the following 2-bit Tunstall code for the alphabet {A, B}. How can we encode the sequence: AAABAABAABAABAAA?

| Sequence | Codeword |
|---|---|
| AAA | 00 |
| AAB | 01 |
| AB | 10 |
| B | 11 |

8 – Generate the codewords using the LZ77 algorithm for the following example:



| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | a | c | a | a |
| | | | | | | | | | a | c | a | a | c |
| | | | | | | | | a | c | a | a | c | a |
| | | | | | | a | c | a | a | c | a | b | c |
| | | | a | c | a | a | c | a | b | c | a | b | a |

9 – Encode the message “aaabaaadaabaado” using LZ78 (representing the dictionary, the indexes, and the codewords).

10 - Encode the message “aaabaaadaabaado” using LZW (represent the original dictionary considering that we have only the lowercase letters of the English dictionary.