**Mobile Programming and Multimedia - exercise on lossless compression**

Amista’ Michael - 2122865

**Exercise**

Encode the string: abcabcabcabcffffffffffff000000000000ffffffffffffffffffffffff

using:

* the LZW algorithm and
* choose an algorithm between Shannon-Fano and Huffman.

Compare the two results in terms of compression ratio.

**LZW algorithm**

Here it is possible to observe the execution, step by step, of LZW encoding algorithm on the given sequence through the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **w** | **k** | **output** | **code** | **symbol** |
| NULL | a |  |  |  |
|  | b |  |  |  |
|  | c |  |  |  |
|  | a |  |  |  |
|  | b |  |  |  |
|  | c |  |  |  |
|  | a |  |  |  |
|  | b |  |  |  |
|  | c |  |  |  |
|  | a |  |  |  |
|  | b |  |  |  |
|  | c |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | 0 |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | f |  |  |  |
|  | EOF |  |  |  |

# bits used to represent the encoded sequence = … bits

(since 8 bits are not enough to represent the encoded sequence which exceeds the bound [0,255] it is necessary to use 2 bytes instead of 1, so we can represent each encoded symbol with 2 bytes = 16 bits)

**Compression ratio** = uncompressed information / compressed information

**Shannon-Fano algorithm**

Here it is possible to observe the tree, resulting from the execution of Shannon-Fano’s algorithm on the given sequence. Each node is characterized by the total number of occurrences for the characters in the set {…}, initially ordered in a descendent way. The set is recursively divided in two portions in a way to obtain balanced divisions of the set with more or less the same number of occurrences.

Immagine che contiene diagramma, linea

Descrizione generata automaticamente

The following table presents the algorithm results, with encodings of every character.

|  |  |
| --- | --- |
| **Character** | **Code** |
| f | 0 |
| 0 | 10 |
| a | 110 |
| b | 1110 |
| c | 1111 |

# bits used to represent the encoded sequence = 104 bits

Since the above table is crucial to decode any output sequence of Shannon-Fano’s algorithm, the table must be contained in the compressed result of the algorithm and due to this it should be considered in the total amount of bits used to represent the Shannon-Fano’s output sequence.

# bits used to represent the table = … bits

# total bits used to represent Shannon-Fano’s output sequence (including the table) = … bits

**Compression ratio** = uncompressed information / compressed information

**Conclusions**

…