Global Journal of Engineering Science and Research Management

LEMPEL - ZIV - WELCH & HUFFMAN” - THE LOSSLESS COMPRESSION TECHNIQUES; (IMPLEMENTATION ANALYSIS AND COMPARISON THEREOF)

Kapil Kapoor*, Dr. Abhay Sharma
* Research Scholar, Sainath University*, Associate Professor, LRIET, Solan, India

DOI: 10.5281/zenodo.55306

KEYWORDS: Gray levels, Histogram, Redundancy, Lempel- Ziv -Welch, Huffman

ABSTRACT

This paper is about the Implementation Analysis and Comparison of Lossless Compression Techniques viz. Lempel-Ziv-Welch and Huffman. LZW technique assigns fixed length code words. It requires no prior information about the probability of occurrence of symbols to be encoded. Basic idea in Huffman technique is that different gray levels occur with different probability (non-uniform- '•histogram). It uses shorter code words for the more common gray levels and longer code words for the less common gray levels. The comparison of Huffman and LZW based on their compression ratio, time, and histogram characteristics.

INTRODUCTION

The authors (s) in this paper are trying to analyze and compare the two commonly used compression techniques, **“LEMPEL - ZIV - WELCH & HUFFMAN”**, according to their algorithms, implementations. As the LEMPEL - ZIV - WELCH commonly called LZW assigns fixed length code words and requires no information about the symbols to be encoded. Where as in the **“HUFFMAN”** coding technique uses different gray levels with different probabilities and it uses short code words for common grey levels and longer code words for less common grey levels. The comparison is being done on the basis of their compression ratio, time, and histogram characteristics.

Lossless Compression Techniques

Lossless compression techniques composed of two independent operations.

- Devising an alternative representation of an image in which inter pixel redundancies are reduced.
- Coding the representation to eliminate coding redundancies.

The Lempel-Ziv-Welch Coding Technique:

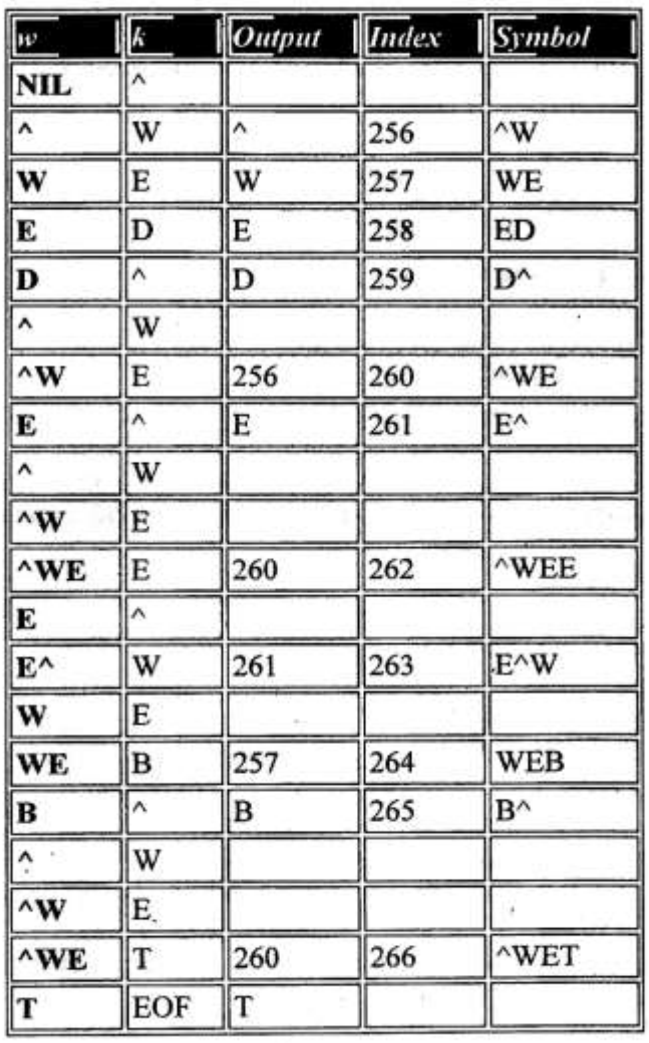
- Assigns fixed length code words.
- Requires no priori information about the probability of occurrence of symbols to be encoded.
- LZW_ starts with a 4K dictionary, of which entries 0-255 refer to individual bytes, and entries 256-4095 refer to substrings. Each time a new code is generated it means a new string has been parsed. New strings are generated by appending the current character K to the end of an existing string w.

LZW Compression Algorithm

```
w = NIL;
while (read a character k)
{if wk exists in the dictionary w = wk;
else
add wk to the dictionary;
output the code for w;
w = k;}
```

Note: Original LZW used dictionary with 4K entries first 256 (0-255) are ASCII codes,

Example: Input string is "**^WED^WE^WEE^WEB^WET**".



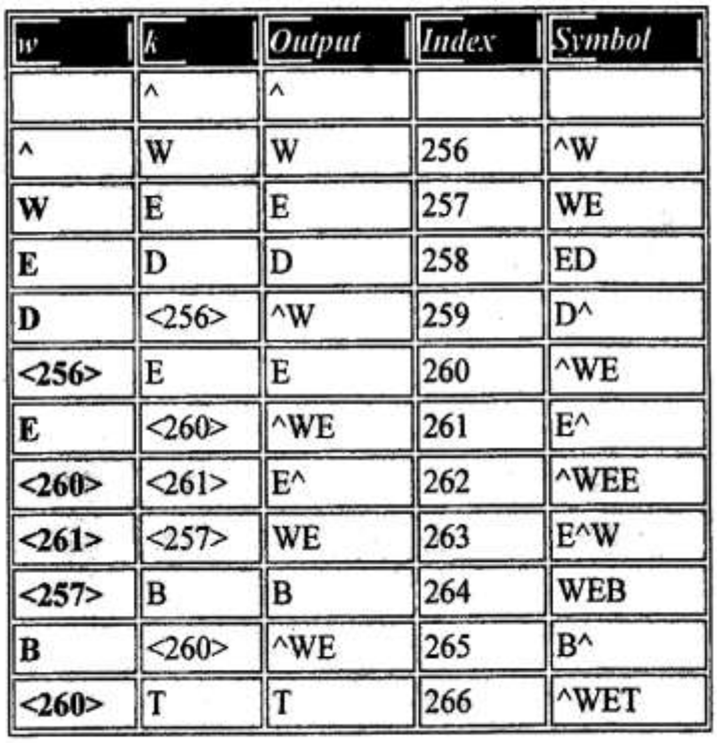
w	*k*	*Output*	*Index*	*Symbol*
NIL	^			
^	W	^	256	^W
W	E	W	257	WE
E	D	E	258	ED
D	^	D	259	D^
^	W			
^W	E	256	260	^WE
E	^	E	261	E^
^	W			
^W	E			
^WE	E	260	262	^WEE
E	^			
E^	W	261	263	E^W
W	E			
WE	B	257	264	WEB
B	^	B	265	B^
^	W			
^W	E			
^WE	T	260	266	^WET
T	EOF	T		

- A 19-symbol input has been reduced to 7-symbol plus 5-code output. Each code/symbol will need more than 8 bits, say 9 bits.
- Usually, compression doesn't start until a large number of bytes (e.g., > 100) are read in.

LZW Decompression Algorithm:

```
read a character k;
output k;
w = k;
while (read a character k) /* k could be a character or a code. */
{
entry = dictionary entry for k;
output entry;
add w +·entry [O) to dictionary;
w=entry;
}
```

Example: Input string is **^WED<256>260><261><257>B<260>T"**



w	*k*	*Output*	*Index*	*Symbol*
	^	^		
^	W	W	256	^W
W	E	E	257	WE
E	D	D	258	ED
D	<256>	^W	259	D^
<256>	E	E	260	^WE
E	<260>	^WE	261	E^
<260>	<261>	E^	262	^WEE
<261>	<257>	WE	263	E^W
<257>	B	B	264	WEB
B	<260>	^WE	265	B^
<260>	T	T	266	^WET

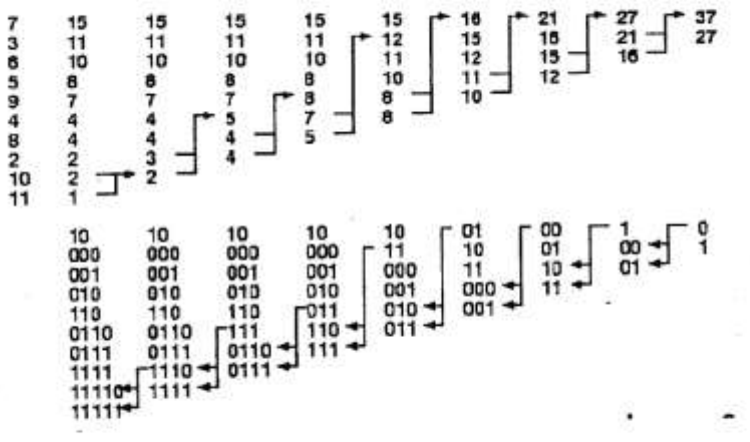
Implementation Analysis of Lempel- Ziv -Welch Technique:
LZW can be made really fast; it grabs a fixed number of bits from input stream, so bit parsing is very easy. Table lookup is automatic.

- Huffman maps fixed length symbols to variable length codes. Optimal only when symbol probabilities are powers of 2.
- Lempel- Ziv-Welch is a dictionary-based compression method. It maps a variable number of symbols to a fixed length code.
- Adaptive algorithms do not need a priori estimation of probabilities; they are more useful in real applications.
- The most remarkable feature of this type of compression is that the entire dictionary has been transmitted to the decoder without actually explicitly transmitting the dictionary. At the end of the run, the decoder will have a dictionary identical to the one the encoder has, built up entirely as part of the decoding process.
- Problem: What if we run out of dictionary space?
  - Solution 1: Keep track of unused entries and use LRU (Least Recently Used).
  - Solution 2: Monitor compression performance and flush dictionary when performance is poor.

The Huffman coding Technique:
Huffman coding yields the smallest possible number of unique code symbols per source symbol.

- Step l .
  1. Sort the gray levels by decreasing probability
  2. Add the two smallest probabilities
  3. Sort the new value into the list, repeat until only two probabilities remain.
- Step 2
  1. Give the code 0 to the highest probability and the code l to the lowest probability in the present node
  2. Go backwards through the tree and add 0 to the highest and 1 to the lowest probability in each node until all gray levels have a unique code.

Huffman Code of an Image:



Implementation Analysis of Huffman technique:

- Algorithm is easy to implement
- Produce a loss less compression of images

  But...........
- Efficiency depends on the accuracy of the statistical model used and type of image. "The algorithm varies with different formats, but few get better than 8:1 compression."
- Compression of image files that contain long runs of identical pixels by Huffman is not as efficient when compared to RLE.
- The Huffman encoding process is usually done in two passes. During the first pass, a statistical model is built, and then in the second pass the image data is encoded based on the generated model. From here we can see that Huffman encoding is a relatively slow process as time is required to build the statistical model in order to archive an efficient compression rate. .
- Another disadvantage of Huffman is that, all codes of the encoded data are of different sizes (not of fixed length). Therefore it is very difficult for the decoder to know that it has reached the last bit of a code, and the only way for it to know is by following the paths of the up-side down tree and coming to an end of it (one of the branch). Thus, if the encoded data is corrupted with additional bits added or bits missing, then whatever that is decoded will be wrong values, and the final image displayed will be garbage.

Comparison of Huffman and LZW:

- Huffman coding provides compression ratio about 36% whereas LZW provides compression ratio up to 60 - 80 %.
- Histogram for Huffman coding is non-uniform whereas in case of LZW its uniform as it uses fixed length codes.
- Huffman coding becomes complex if symbols increases, as if J symbols then J-2 reductions are needed whereas LZW starts with a 4K dictionary, of which entries 0-255 refer to individual bytes, and entries 256-4095 refer to substrings.
- Huffman uses VLC (variable length coding), whereas LZW uses fixed length codes.
- MPEG-2 standard uses Huffman Coding whereas LZW is used in GIFF, TIFF, and JPEG standard.
- Huffman coding requires the information about probability of occurrence of symbols whereas LZW technique does not requires statistical information about probabilities of each symbols.

CONCLUSION AND DISCUSSION

- LZW is really fast; it grabs a fixed number of bits from input stream, so bit parsing is very easy. Table lookup is automatic.
- Huffman maps fixed length symbols to variable length codes. Optimal only when symbol probabilities are powers of 2.

- Lempel-Ziv-Welch is a dictionary-based compression method. It maps a variable number of symbols to a fixed length code.
- Adaptive algorithms do not need a priori estimation of probabilities; they are more useful in real applications.
- The most remarkable feature of LZW is that the entire dictionary has been transmitted to the decoder without actually explicitly transmitting the dictionary. At the end of the run, the decoder will have a dictionary identical to the one the encoder has, built up entirely as part of the decoding process. ·
- Huffman coding requires the information about probability of occurrence of symbols whereas LZW technique does not requires statistical information about probabilities of each symbols.
- Huffman coding becomes complex if symbols increases, as if J symbols then J-2 reductions are needed whereas LZW starts with a 4K dictionary, of hitch entries 0-255 refer to individual bytes, and entries 256-4095 refer to substrings.
- Huffman uses VLC (variable length coding), whereas LZW uses fixed length codes.
- MPEG-2 standard uses Huffman Coding whereas LZW is used in GIFF, TIFF and JPEG standard.

REFERENCES

1. "Gonzalez and Woods ", *Digital Image Processing* by Prentice Hall.
2. "Hall, E.L. ". Computer Image Processing and Recognition.
3. "Kenneth R. Castleman", Digital Image Processing