Dictionary Coding

- Does not use statistical knowledge of data.
- Encoder: As the input is processed develop a dictionary and transmit the index of strings found in the dictionary.
- Decoder: As the code is processed reconstruct the dictionary to invert the process of encoding.
- Examples: LZW, LZ77, Sequitur,
- Applications: Unix Compress, gzip, GIF

LZW Encoding Algorithm

Repeat
find the longest match w in the dictionary
output the index of w
put wa in the dictionary where a was the unmatched symbol

LZW Encoding Example (1)

Dictionary

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input: a b a b a b a

Output:

0 1

a
b

LZW Encoding Example (2)

Dictionary

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ab</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input: a b a b a b a

Output:

0 1 2

a
b
ab

LZW Encoding Example (3)

Dictionary

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ab</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ba</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input: a b a b a b a

Output:

0 1 2 3

a
b
ab
ba
LZW Encoding Example (4)

Dictionary
0 a
1 b
2 ab
3 ba
4 aba

LZW Encoding Example (5)

Dictionary
0 a
1 b
2 ab
3 ba
4 aba
5 abab

LZW Encoding Example (6)

Dictionary
0 abababa
1 a
2 b
3 ab
4 aba
5 abab

LZW Decoding Algorithm

- Emulate the encoder in building the dictionary. Decoder is slightly behind the encoder.

Initialize dictionary;
decode first index to w;
put w? in dictionary;
repeat
decode the first symbol s of the index;
complete the previous dictionary entry with s;
finish decoding the remainder of the index;
put w? in the dictionary where w was just decoded;

LZW Decoding Example (1)

Dictionary
0 a
1 b
2 a?

LZW Decoding Example (2a)

Dictionary
0 a
1 b
2 ab
LZW Decoding Example (2b)

Dictionary
0  a
1  b
2  ab
3  b?

0  a
1  b
2  ab
3  b?

LZW Decoding Example (3a)

Dictionary
0  a
1  b
2  ab
3  ba

0  a
1  b
2  ab
3  ba

LZW Decoding Example (3b)

Dictionary
0  a
1  b
2  ab
3  ba
4  ab?

0  a
1  b
2  ab
3  ba
4  ab?

LZW Decoding Example (4a)

Dictionary
0  a
1  b
2  ab
3  ba
4  ab?

0  a
1  b
2  ab
3  ba
4  ab?

LZW Decoding Example (4b)

Dictionary
0  a
1  b
2  ab
3  ba
4  ab?
5  aba?

0  a
1  b
2  ab
3  ba
4  ab?
5  aba?

LZW Decoding Example (5a)

Dictionary
0  a
1  b
2  ab
3  ba
4  ab?
5  abab

0  a
1  b
2  ab
3  ba
4  ab?
5  abab
LZW Decoding Example (5b)

Dictionary
0  a
1  b
2  ab
3  ba
4  aba
5  abab
6  ba?

LZW Decoding Example (6a)

Dictionary
0  a
1  b
2  ab
3  ba
4  aba
5  abab
6  bab

LZW Decoding Example (6b)

Dictionary
0  a
1  b
2  ab
3  ba
4  aba
5  abab
6  bab
7  bab?

Trie Data Structure for Encoder's Dictionary

- Fredkin (1960)

Encoder Uses a Trie (1)

Encoder Uses a Trie (2)
**Decoder’s Data Structure**

- Simply an array of strings:

  - a, c
  - b, d
  - c, a
  - d, br
  - r, ab
  - ab, r?
  - br, r
  - ra, a
  - ac

  - 0 1 4 0 2 0 3 5 7 12 8 ...
  - abracadabra abracadabra

**Bounded Size Dictionary**

- Bounded Size Dictionary
  - n bits of index allows a dictionary of size $2^n$
  - Doubtful that long entries in the dictionary will be useful.

- Strategies when the dictionary reaches its limit:
  1. Don’t add more, just use what is there.
  2. Throw it away and start a new dictionary.
  3. Double the dictionary, adding one more bit to indices.
  4. Throw out the least recently visited entry to make room for the new entry.

**Notes on LZW**

- Extremely effective when there are repeated patterns in the data that are widely spread.
- Negative: Creates entries in the dictionary that may never be used.
- Applications:
  - Unix compress, GIF, V.42 bis modem standard