Data Compression

- Reduce the size of data.
  - Reduces storage space and hence storage cost.
  - Compression ratio = original data size/compressed data size
  - Reduces time to retrieve and transmit data.

Lossless And Lossy Compression

- compressedData = compress(originalData)
- decompressedData = decompress(compressedData)
- When originalData = decompressedData, the compression is lossless.
- When originalData != decompressedData, the compression is lossy.

Lossless And Lossy Compression

- Lossy compressors generally obtain much higher compression ratios than do lossless compressors.
  - Say 100 vs. 2.
- Lossless compression is essential in applications such as text file compression.
- Lossy compression is acceptable in many imaging applications.
  - In video transmission, a slight loss in the transmitted video is not noticed by the human eye.

Text Compression

- Lossless compression is essential.
- Popular text compressors such as zip and Unix’s compress are based on the LZW (Lempel-Ziv-Welch) method.

LZW Compression

- Character sequences in the original text are replaced by codes that are dynamically determined.
- The code table is not encoded into the compressed text, because it may be reconstructed from the compressed text during decompression.

LZW Compression

- Assume the letters in the text are limited to \{a, b\}.
  - In practice, the alphabet may be the 256 character ASCII set.
  - The characters in the alphabet are assigned code numbers beginning at 0.
- The initial code table is:
  
<table>
<thead>
<tr>
<th>code</th>
<th>key</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
</tr>
</tbody>
</table>
LZW Compression

- Original text = abababbabaabbbaabba
- Compression is done by scanning the original text from left to right.
- Find longest prefix p for which there is a code in the code table.
- Represent p by its code pCode and assign the next available code number to pc, where c is the next character in the text that is to be compressed.

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

• p = a
• pCode = 0
• c = b
• Represent a by 0 and enter ab into the code table.
• Compressed text = 0

LZW Compression

- Original text = abababbabaabbbaabbaabba
- Compressed text = 01
- p = ab
- pCode = 2
- c = a
- Represent ab by 2 and enter aba into the code table.
• Compressed text = 012

LZW Compression

- Original text = abababbabaabbbaabbaabba
- Compressed text = 012
- p = ab
- pCode = 2
- c = b
- Represent ab by 2 and enter abb into the code table.
• Compressed text = 0122
**LZW Compression**

- Original text = `ababababaabbabbaabba`
- Compressed text = `012233`  
- `p = ba`
- `pCode = 3`
- `c = a`
- Represent `ba` by `3` and enter `baa` into the code table.
- Compressed text = `012233`  

**LZW Compression**

- Original text = `ababababaabbabbaabba`
- Compressed text = `012233`  
- `p = abb`
- `pCode = 5`
- `c = a`
- Represent `abb` by `5` and enter `abba` into the code table.
- Compressed text = `01223358`  

**LZW Compression**

- Original text = `ababababaabbabbaabbaabba`
- Compressed text = `01223358`  
- `p = abba`
- `pCode = 8`
- `c = a`
- Represent `abba` by `8` and enter `abbaa` into the code table.
- Compressed text = `01223358`  

**Code Table Representation**

- Dictionary.
  - Pairs are (key, element) = (key, code).
  - Operations are : get(key) and put(key, code)
  - Limit number of codes to $2^{12}$.
  - Use a hash table.
  - Convert variable length keys into fixed length keys.
  - Each key has the form pc, where the string p is a key that is already in the table.
  - Replace pc with (pCode)c.

**Code Table Representation**

- Dictionary.
  - Pairs are (key, element) = (key, code).
  - Operations are : get(key) and put(key, code)
  - Limit number of codes to $2^{12}$.
  - Use a hash table.
  - Convert variable length keys into fixed length keys.
  - Each key has the form pc, where the string p is a key that is already in the table.
  - Replace pc with (pCode)c.
### LZW Decompression

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

- Original text = `ababbbabbaabbababba`
- Compressed text = `012233588`
- Convert codes to text from left to right.
- 0 represents `a`.
- Decompressed text = `a`
- `pCode = 0` and `p = a`.
- `p = a` followed by next text character (c) is entered into the code table.

### LZW Decompression

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

- Original text = `ababbbabbaabbababba`
- Compressed text = `012233588`
- 1 represents `b`.
- Decompressed text = `ab`
- `pCode = 1` and `p = b`.
- `lastP = a` followed by first character of `p` is entered into the code table.

### LZW Decompression

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

- Original text = `ababbaaabbaabbabbaabba`
- Compressed text = `012233588`
- 2 represents `ab`.
- Decompressed text = `abab`
- `pCode = 2` and `p = ab`.
- `lastP = b` followed by first character of `p` is entered into the code table.

### LZW Decompression

<table>
<thead>
<tr>
<th>Code</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>ab</td>
</tr>
<tr>
<td>3</td>
<td>ba</td>
</tr>
<tr>
<td>4</td>
<td>aba</td>
</tr>
</tbody>
</table>

- Original text = `ababbaaabbaabbabbaabba`
- Compressed text = `012233588`
- 3 represents `ba`.
- Decompressed text = `abababa`
- `pCode = 3` and `p = ba`.
- `lastP = ba` followed by first character of `p` is entered into the code table.
LZW Decompression

- Original text = abababbabaabbbaabbbaabbbaabbbaabba
- Compressed text = 012233588
- 5 represents abb
- Decompressed text = abababbabaabbbaabbbaaab
  - pCode = 5 and p = abb.
  - lastP = ba followed by first character of p is entered into the code table.

LZW Decompression

- Original text = abababbabaabbbaabbbaabbbaabbbaabbbaabba
- Compressed text = 012233588
- 8 represents abba
- When a code is not in the table, its key is lastP followed by first character of lastP.
  - lastP = abb
  - So 8 represents abba.

Code Table Representation

- Dictionary.
  - Pairs are (key, element) = (code, what the code represents) = (code, codeKey).
  - Operations are: get(key) and put(key, code).
  - Keys are integers 0, 1, 2, ...
  - Use a 1D array codeTable.
    - codeTable[code] = codeKey.
    - Each code key has the form pc, where the string p is a code key that is already in the table.
    - Replace pc with (pCode)c.

Time Complexity

- Compression.
  - \(O(n)\) expected time, where \(n\) is the length of the text that is being compressed.
- Decompression.
  - \(O(n)\) time, where \(n\) is the length of the decompressed text.