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

- \( \text{compressedData} = \text{compress(originalData)} \)
- \( \text{decompressedData} = \text{decompress(compressedData)} \)
- When \( \text{originalData} = \text{decompressedData} \), the compression is lossless.
- When \( \text{originalData} \neq \text{decompressedData} \), 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>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>
LZW Compression

- Original text = abababbabaabbabbaabba
- 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 $p\text{Code}$ 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>

- Original text = abababbabaabbabbaabba
- $p = a$
- $p\text{Code} = 0$
- $c = b$
- Represent $a$ by 0 and enter $ab$ into the code table.
- Compressed text = 0
**LZW Compression**

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

- Original text = abababbabaabbabbaabba
- Compressed text = 0
- p = b
- pCode = 1
- c = a
- Represent b by 1 and enter ba into the code table.
- Compressed text = 01

---

**LZW Compression**

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

- Original text = abababbabaabbabbaabbaabba
- 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

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
</tr>
</tbody>
</table>

- Original text = `abababbabaabbabbaabba`  
- 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

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
<td>bab</td>
</tr>
</tbody>
</table>

- Original text = `ababababbabaabbabbaabbaabba`  
- Compressed text = `0122`  
- p = ba  
- pCode = 3  
- c = b  
- Represent ba by 3 and enter bab into the code table.  
- Compressed text = `01223`
LZW Compression

- Original text = abababababaabbababaabba
- Compressed text = 01223
- p = ba
- pCode = 3
- c = a
- Represent ba by 3 and enter baa into the code table.
- Compressed text = 012233

LZW Compression

- Original text = abababababaabbababaabba
- Compressed text = 012233
- p = abb
- pCode = 5
- c = a
- Represent abb by 5 and enter abba into the code table.
- Compressed text = 0122335
### LZW Compression

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
<td>bab</td>
<td>baa</td>
<td>abbaabbba</td>
<td></td>
</tr>
</tbody>
</table>

- Original text = ababababaabbabbaabbba
- Compressed text = 0122335
- p = abba
- pCode = 8
- c = a
- Represent abba by 8 and enter abbaa into the code table.
- Compressed text = 01223358

### LZW Compression

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
<td>bab</td>
<td>baa</td>
<td>abbaabbba</td>
<td></td>
</tr>
</tbody>
</table>

- Original text = ababababaabbabbaabbbaabba
- Compressed text = 01223358
- p = abba
- pCode = 8
- c = null
- Represent abba by 8.
- Compressed text = 012233588
• 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**

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
<td>bab</td>
<td>baa</td>
<td>abba</td>
<td>abbaabbaa</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>0b</td>
<td>1a</td>
<td>2a</td>
<td>2b</td>
<td>3b</td>
<td>3a</td>
<td>5a</td>
<td>8a</td>
</tr>
</tbody>
</table>
LZW Decompression

<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>

- Original text = ababababaabbabbaabba
- 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.

### Code Table

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

- Original text = ababababaabbabbaabbaabba
- 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>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
</tr>
</tbody>
</table>

- Original text = abababbabaabbabbaaba
- 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>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
</tr>
</tbody>
</table>

- Original text = ababababababbaabbabaabba
- Compressed text = 012233588
- 3 represents ba
- Decompressed text = ababbaa.
- pCode = 3 and p = ba.
- lastP = ab followed by first character of p is entered into the code table.

LZW Decompression

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
<td>bab</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
<td>bab</td>
<td>baa</td>
</tr>
</tbody>
</table>

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

When a code is not in the table, its key is lastP followed by first character of lastP.

- lastP = abb
- So 8 represents abba.
LZW Decompression

- Original text = abababbabaabbabbaabba
- Compressed text = 01223358
- 8 represents abba
- Decompressed text = abababbabaabbabbaabba.
- pCode = 8 and p = abba.
- lastP = abba followed by first character of p is entered into the code table.

<table>
<thead>
<tr>
<th>code</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>a</td>
<td>b</td>
<td>ab</td>
<td>ba</td>
<td>aba</td>
<td>abb</td>
<td>bab</td>
<td>baa</td>
<td>abba</td>
<td>abba</td>
</tr>
</tbody>
</table>

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.