Dictionary Coding
LZ77

The Dictionary is Implicit

- Ziv and Lempel, 1977
- Use the string coded so far as a dictionary.
- Given that $x_1, x_2, ..., x_n$ has been coded we want to code $x_{n+1}, x_{n+2}, ..., x_{n+k}$ for the largest $k$ possible.

Solution A

- If $x_{n+1}, x_{n+2}, ..., x_{n+k}$ is a substring of $x_1, x_2, ..., x_n$ then $x_{n+1}, x_{n+2}, ..., x_{n+k}$ can be coded by $\langle j, k \rangle$ where $j$ is the beginning of the match.
- Example
  
  abababababababab....
  coded
  ababababababababababababab....
  $\langle 2, 8 \rangle$

Solution A Problem

- What if there is no match at all in the dictionary?
- Solution B. Send tuples $\langle j, k, x \rangle$ where
  - If $k = 0$ then $x$ is the unmatched symbol
  - If $k > 0$ then the match starts at $j$ and is $k$ long and the unmatched symbol is $x$.

Solution B

- If $x_{n+1}, x_{n+2}, ..., x_{n+k}$ is a substring of $x_1, x_2, ..., x_n$ and $x_{n+1}, x_{n+2}, ..., x_{n+k}$ is not then $x_{n+1}, x_{n+2}, ..., x_{n+k}$ $x_{n+k+1}$ can be coded by $\langle j, k, x_{n+k+1} \rangle$
  where $j$ is the beginning of the match.
- Example
  
  ababababababababababababab....
  ababababababababababababababababababab....
  $\langle 0, 0, a \rangle$
  $\langle 0, 0, b \rangle$

Solution B Example

- $a$ bababababababababababababab....
  $\langle 0, 0, a \rangle$
- $b$ bababababababababababababab....
  $\langle 0, 0, b \rangle$
- $a$ baba bababababababababababababababababababababab....
  $\langle 1, 10, a \rangle$
**Surprise Code!**

\[ \text{a bababababababababab}$\]
\[<0,0,a>\]
\[\text{a b bababababababababab}$\]
\[<0,0,b>\]
\[\text{a b babababababababababab}$\]
\[<1,22,8>\]

**Surprise Decoding**

\[ <0,0,a><0,0,b><1,22,8>\]
\[<0,0,a> a \]
\[<0,0,b> b \]
\[<1,22,8> a \]
\[<2,21,8> b \]
\[<3,20,8> a \]
\[<4,19,8> b \]
\[<22,1,8> b \]
\[<23,0,8> $ \]

**Solution C**

- The matching string can include part of itself!
- If \( x_{n+1}x_{n+2}...x_{n+k} \) is a substring of \( x_1x_2...x_nx_{n+1}x_{n+2}...x_{n+k} \) that begins at \( j \leq n \) and \( x_{n+1}x_{n+2}...x_{n+k}x_{n+k+1} \) is not then \( x_{n+1}x_{n+2}...x_{n+k}x_{n+k+1} \) can be coded by \( <j,k,x_{n+k+1}> \).

**In Class Exercise**

- Use Solution C to code the string
  - abaababaaaab$

**Bounded Buffer – Sliding Window**

- We want the triples \( <j,k,x> \) to be of bounded size. To achieve this we use bounded buffers.
  - Search buffer of size \( s \) is the symbols \( x_{n-s+1}...x_n \).
    - Match pointer can start in search buffer and go into the look-ahead buffer but no farther.
  - Look-ahead buffer of size \( t \) is the symbols \( x_{n+1}...x_{n+t} \).

![Sliding Window Diagram]

- Tuple

\[<2,5,a>\]
Search in the Sliding Window

<table>
<thead>
<tr>
<th></th>
<th>Offset</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaababaaab$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>aaababaaab$</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>aaababaaab$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>aaababaaab$</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>aaababaaab$</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>aaababaaab$</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Coding Example

\[ s = 4, t = 4, a = 3 \]

<table>
<thead>
<tr>
<th>Tuple</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4, 5, a)</td>
</tr>
</tbody>
</table>

Coded Example

\[ \text{tuple} \]

<table>
<thead>
<tr>
<th>Tuple</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0, a)</td>
</tr>
<tr>
<td>(1, 3, b)</td>
</tr>
<tr>
<td>(2, 5, a)</td>
</tr>
<tr>
<td>(4, 2, $)</td>
</tr>
</tbody>
</table>

Coding the Tuples

- Simple fixed length code
  \[ [\log_2(s+1)] + [\log_2(s+t+1)] + [\log_2(a)] \]
  \[ s = 4, t = 4, a = 3 \]
  tuple fixed code
  \[ (2, 5, a) \] 010 0101 00

- Variable length code using adaptive Huffman or arithmetic code on Tuples
  - Two passes, first to create the tuples, second to code the tuples
  - One pass, by pipelining tuples into a variable length coder

Zip and Gzip

- Search Window
  - Search buffer 32KB
  - Look-ahead buffer 258 Bytes
- How to store such a large dictionary
  - Hash table that stores the starting positions for all three byte sequences.
  - Hash table uses chaining with newest entries at the beginning of the chain. Stale entries can be ignored.
- Second pass for Huffman coding of tuples.
- Coding done in blocks to avoid disk accesses.
Notes on LZ77

- Very popular especially in unix world
- Many variants and implementations
  - Zip, Gzip, PNG, PKZip, Lharc, ARJ
- Tends to work better than LZW
  - LZW has dictionary entries that are never used
  - LZW has past strings that are not in the dictionary
  - LZ77 has an implicit dictionary. Common tuples are coded with few bits.