Lempel–Zev

Drawbacks of Huffman's?
- must read entire file before encoding
- encodes only single characters (not strings)

ex: AAA BAAA

Huffman's

```
  o
 A/ B
```

⇒ A = 0 B = 1
encoded: 0001000

Better: AAA = 0 B = 1 encoded: 010

Also, Huffman encodes: char → binary
In practice: char → ASCII → binary.

Assume: have ASCII table (code for all 256 characters)

Goals:
- encode text while reading
- encode strings instead of chars.

"sliding window" approach

ex: the cat the hat

With fixed-length sliding window of size 2
encode: th, he, ec, ... (th already encoded) the, heh, eh...
But we want longer strings (not fixed-length)!

So, start with window of size 2 (all strings of length 1 already have ASCII codes) and extend the window.

When should we extend?
⇒ When the current string has already been encoded

ex: the cat, the hat

```
th - encode
he - "
ec - "
```

th - already encoded, extend window, encode "the"
en - encode

Frequently checking whether a string has a code
How to do this quickly?
Store strings, codes in a hash table/dictionary

Recall: All single chars are already in the table
ex: A B B A B D B A B B A
newstr code
AB 256
BB 257
BA 258
ABB 259
BBA 260
ABB A 261

output: 65 66 66 256 257 259 65

A
Lempel-Ziv (msg, dictionary)

str = msg.getNextChar()  // first char (in Dictionary)

While (there are still chars in msg):

  c = msg.getNextChar()

  if (dictionary.contains (str + c))
    str = str + c  // build a longer string
    // (extend window length)
  else

    output str code  // longest string with a code
    dictionary.add (str + c)
    str = c  // slide the window

  // after while

output str code

How good is LZ?

No compression: 8 bits * 11 chars = 88 bits
Compressed w/LZ: (8 bits) * 4 + (9 bits) * 3 = 59 bits
$\Rightarrow 33%$ reduction.

**Effectiveness:** on Alice in Wonderland

Huffman's: 15%

Lempel-Ziv: 50%

**Run Time:** n-length of msg, $1A1 = _{table\ size}O(n)$ if contains is $O(1)$

$O(1A1n)$ if contains searches table (unlikely)

**Another example:**

```
A B R A C A D A B R A  (C = 67, D=68, R = 114)
```

```
newstr  code
--------
AB  256
BR  257
RA  258
AC  259
CA  260
AD  261
```

**Encoding:** 65 66 114 65 67 65 DA 262

AB  256 258

**No compression:** $8(11) = 88$ bits

**Compressed:** $8(7) + 2(9) = 74$ bits
Yet another example:

```
BA  BA  AA
↑  ↑  ↑  ↑  ↑  ↑  ↑
```

<table>
<thead>
<tr>
<th>newstr</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA</td>
<td>256</td>
</tr>
<tr>
<td>AB</td>
<td>257</td>
</tr>
<tr>
<td>BAA</td>
<td>258</td>
</tr>
<tr>
<td>ABA</td>
<td>259</td>
</tr>
<tr>
<td>AA</td>
<td>260</td>
</tr>
</tbody>
</table>

```
output: 66 65 256 257 65 260
```