LZW

It is a dictionary method for compressing. The algorithm is an improvement to the LZ77 method.

Let's start with an example:

- We are going to assume that our dictionary contains only the following "Symbols": A, B, C, D, E
- We initialize the dictionary as follows. This will also be the FIXED part

Index	Symbol
0	A
1	B
2	C
3	D
4	E

- The same dictionary will be used by the decoder as well.
- In your real-life implementation you will have all the symbols (ASCII bytes from 0-255).
- In your real-life implementation the dictionary can be initialize something like

Index	Symbol
0	ASCII char with value 0
1	ASCII char with value 1
2	ASCII char with value 2
3	ASCII char with value 3
4	ASCII char with value 4
…	…
255	ASCII char with value 255

- Let's assume we want to encrypt the following text: ABCDABCDABDCABCDABDDCD

ı	rypt	tionary values	UTPUT
A inside dictionary. Found at index 0.	**A**BCDABCDABDCABCDABDDCD		
AB inside dictionary. Not found. Emit 0(because A was found at index 0). Add AB at the end of dictionary.	**AB**CDABCDABDCABCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB	0
B inside dictionary. Found at index 1.	**B**CDABCDABDCABCDABDDCD		
BC inside dictionary. Not found. Emit 1 (because B was found at index 1). Add BC at the end of dictionary.	**BC**DABCDABDCABCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB 6 BC	1
C inside dictionary. Found at index 2.	**C**DABCDABDCABCDABDDCD		
CD inside dictionary. Not found. Emit 2 (because C was found at index 2). Add CD at the end of dictionary.	**CD**ABCDABDCABCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D	2

		4 E 5 AB 6 BC 7 CD	
D inside dictionary. Found at index 3.	**D**ABCDABDCABCDABDDCD		
DA inside dictionary. Not found. Emit 3 (because D was found at index 3). Add DA at the end of dictionary.	**DA**BCDABDCABCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB 6 BC 7 CD 8 DA	3
A inside dictionary. Found at index 0.	**A**BCDABDCABCDABDDCD		
AB inside dictionary. Found at index 5.	**AB**CDABDCABCDABDDCD		
ABC inside dictionary. Not found. Emit 5 (because AB was found at index 5). Add ABC at the end of dictionary.	**ABC**DABDCABCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB 6 BC 7 CD 8 DA 9 ABC	5
C inside dictionary. Found at index 2.	**C**DABDCABCDABDDCD		
CD inside dictionary. Found at index 7.	**CD**ABDCABCDABDDCD		
CDA inside dictionary. Not found. Emit 7 (because CD was found at index 7). Add CDA at the end of dictionary.	**CDA**BDCABCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB 6 BC 7 CD 8 DA 9 ABC 10 CDA	7
A inside dictionary. Found at index 0.	**A**BDCABCDABDDCD		
AB inside dictionary. Found at index 5.	**AB**DCABCDABDDCD		
ABD inside dictionary. Not found. Emit 5 (because AB was found at index 5). Add ABD at the end of dictionary.	**ABD**CABCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB 6 BC 7 CD	5

		8 DA 9 ABC 10 CDA 11 ADB	
D inside dictionary. Found at index 3.	**D**CABCDABDDCD		
DC inside dictionary. Not found. Emit 3 (because D was found at index 3). Add DC at the end of dictionary.	**DC**ABCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB 6 BC 7 CD 8 DA 9 ABC 10 CDA 11 ADB 12 DC	3
C inside dictionary. Found at index 2.	**C**ABCDABDDCD		
CA inside dictionary. Not found. Emit 2 (because C was found at index 2). Add CA at the end of dictionary.	**CA**BCDABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB 6 BC 7 CD 8 DA 9 ABC 10 CDA 11 ADB 12 DC 13 CA	2
A inside dictionary. Found at index 0.	**A**BCDABDDCD		
AB inside dictionary. Found at index 5.	**AB**CDABDDCD		
ABC inside dictionary. Found at index 9.	**ABC**DABDDCD		
ABCD inside dictionary. Not found. Emit 9 (because ABC was found at index 9). Add ABCD at the end of dictionary.	**ABCD**ABDDCD	Index / Symbol 0 A 1 B 2 C 3 D 4 E 5 AB 6 BC 7 CD 8 DA 9 ABC 10 CDA 11 ADB 12 DC 13 CA 14 ABCD	9
D inside dictionary. Found at index 3.	**D**ABDDCD		

DA inside dictionary. Found at index 8.	**DA**BDDCD		
DAB inside dictionary. Not found. Emit 8 (because DA was found at index 8). Add DAB at the end of dictionary.	**DAB**DDCD	<table><tr><th>Index</th><th>Symbol</th></tr><tr><td>0</td><td>A</td></tr><tr><td>1</td><td>B</td></tr><tr><td>2</td><td>C</td></tr><tr><td>3</td><td>D</td></tr><tr><td>4</td><td>E</td></tr><tr><td>5</td><td>AB</td></tr><tr><td>6</td><td>BC</td></tr><tr><td>7</td><td>CD</td></tr><tr><td>8</td><td>DA</td></tr><tr><td>9</td><td>ABC</td></tr><tr><td>10</td><td>CDA</td></tr><tr><td>11</td><td>ADB</td></tr><tr><td>12</td><td>DC</td></tr><tr><td>13</td><td>CA</td></tr><tr><td>14</td><td>ABCD</td></tr><tr><td>15</td><td>DAB</td></tr></table>	8
B inside dictionary. Found at index 1.	**B**DDCD		
BD inside dictionary. Not found. Emit 1 (because B was found at index 1). Add BD at the end of dictionary.	**BD**DCD	<table><tr><th>Index</th><th>Symbol</th></tr><tr><td>0</td><td>A</td></tr><tr><td>1</td><td>B</td></tr><tr><td>2</td><td>C</td></tr><tr><td>3</td><td>D</td></tr><tr><td>4</td><td>E</td></tr><tr><td>5</td><td>AB</td></tr><tr><td>6</td><td>BC</td></tr><tr><td>7</td><td>CD</td></tr><tr><td>8</td><td>DA</td></tr><tr><td>9</td><td>ABC</td></tr><tr><td>10</td><td>CDA</td></tr><tr><td>11</td><td>ADB</td></tr><tr><td>12</td><td>DC</td></tr><tr><td>13</td><td>CA</td></tr><tr><td>14</td><td>ABCD</td></tr><tr><td>15</td><td>DAB</td></tr><tr><td>16</td><td>DB</td></tr></table>	1
D inside dictionary. Found at index 3.	**D**DCD		
DD inside dictionary. Not found. Emit 3 (because D was found at index 3). Add DD at the end of dictionary.	**DD**CD	<table><tr><th>Index</th><th>Symbol</th></tr><tr><td>0</td><td>A</td></tr><tr><td>1</td><td>B</td></tr><tr><td>2</td><td>C</td></tr><tr><td>3</td><td>D</td></tr><tr><td>4</td><td>E</td></tr><tr><td>5</td><td>AB</td></tr><tr><td>6</td><td>BC</td></tr><tr><td>7</td><td>CD</td></tr><tr><td>8</td><td>DA</td></tr><tr><td>9</td><td>ABC</td></tr><tr><td>10</td><td>CDA</td></tr><tr><td>11</td><td>ADB</td></tr><tr><td>12</td><td>DC</td></tr><tr><td>13</td><td>CA</td></tr><tr><td>14</td><td>ABCD</td></tr><tr><td>15</td><td>DAB</td></tr><tr><td>16</td><td>DB</td></tr><tr><td>17</td><td>DD</td></tr></table>	3
D inside dictionary. Found at index 3.	**D**CD		

DC inside dictionary. Found at index 12.	DCD		
DCD inside dictionary. Not found. Emit 12 (because DC was found at index 12). Add DCD at the end of dictionary.	DCD	<table><tr><th>Index</th><th>Symbol</th></tr><tr><td>0</td><td>A</td></tr><tr><td>1</td><td>B</td></tr><tr><td>2</td><td>C</td></tr><tr><td>3</td><td>D</td></tr><tr><td>4</td><td>E</td></tr><tr><td>5</td><td>AB</td></tr><tr><td>6</td><td>BC</td></tr><tr><td>7</td><td>CD</td></tr><tr><td>8</td><td>DA</td></tr><tr><td>9</td><td>ABC</td></tr><tr><td>10</td><td>CDA</td></tr><tr><td>11</td><td>ADB</td></tr><tr><td>12</td><td>DC</td></tr><tr><td>13</td><td>CA</td></tr><tr><td>14</td><td>ABCD</td></tr><tr><td>15</td><td>DAB</td></tr><tr><td>16</td><td>DB</td></tr><tr><td>17</td><td>DD</td></tr><tr><td>18</td><td>DCD</td></tr></table>	12
D inside dictionary. Found at index 3.	D		3

- To summarize we will write into the "encoded" file the following **INDEXES**: 0,1,2,3,5,7,5,3,2,9,8,1,3,12,3
- The algorithm for encoding can be "summarized" into something like:

```
string s;char ch;
s = empty string;
while (there is still data to be read)
{
    ch = read a character;
    if (dictionary contains s+ch)
    {
            s = s+ch;
    }
    else
    {
            encode s to output file;
            add s+ch to dictionary;
            s = ch;
    }
}
encode s to output file;
```

Implementation details for **Coder**:

- UI shall contain the following option
  - Load button – it will offer the user to load a file and will display the full file path (including name) for the selected file
  - Choose the number of bits on which to write the INDEX value from 9-15
  - Choose what to do when Dictionary is full: freeze OR empty
    - If you choose 9 (you will write the indexes on 9 bits) => the dictionary can contain the maximum 2^9-1 values.
    - The question appears what to do when the dictionary gets full
      - Freeze – you will not add inside the dictionary after it's full– you'll use the existing values
      - Empty – you will empty the dictionary (but keeping the FIXED part – meaning the first 256 positions there)
  - Encoded button – when press it will create another file with the following filename format:
    - File.ext. [f OR e – Freeze or Emtpty]l[index_number_selected_from_UI].LZW
  - A checkbox to display on UI the emitted indexes

- The encoded file will contain the following
    - The first 4 bits will contain the index bit size selected from UI
    - The next bit will be 1 bit: 1 if method is FREEZE/0 if method is empty
    - You will then read from the input file and emit INDEXES in the encoded file:
        - Each index you will write on the number of bits selected from UI
    - At the end you'll write additionally 7 bits of "1" to fill the BitWriter buffer

Implementation details for **Decoder**

- UI shall contain the following options
    - Load button – it will offer to the user to load a file and will display the full file path (including name) for the selected file. The file must have the LZW extention
    - Decode button – when press it will create another file with the following filename format:
        - File.ext. [f OR e – Freeze or Emtpty]l[index_number_selected_from_UI].LZW.ext
    - The decoder shall read the following from the input encoded file
        - 4 bits – the value of the index bit size (let's call it X)
        - 1 bits – the value meaning what to do when the dictionary is full (1 Freeze / 0 empty)
    - You will than keep reading (until you arrive to the end of file) X bits from the input file, or the remaining bits inside the file is less than X