LZW compression

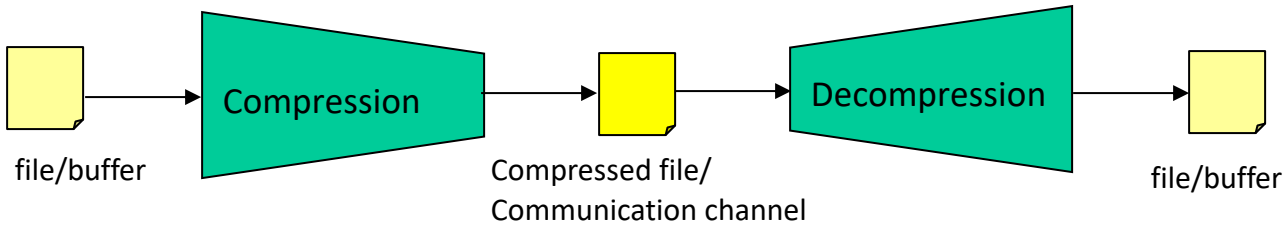
Injung Kim modified by Nannan Wen

10/11/2019

Lempel-Ziv-Welch (LZW) Algorithm

Preliminaries:

- A dictionary that is indexed by “codes” is used.
- The dictionary is assumed to be initialized with 256 entries (indexed with ASCII codes 0 through 255) representing the ASCII table.
- The compression algorithm assumes that the output is either a file or a communication channel. The input being a file or buffer.
- Conversely, the decompression algorithm assumes that the input is a file or a communication channel and the output is a file or a buffer.



LZW Compression

Example: abcabcabcd

Output:

W:

```
set w = NIL
 loop
        read a character k
        if wk exists in the dictionary
              w = wk
        else
              output the code for w
              add wk to the dictionary
              w = k
endloop
```

w	code
a	1
b	2
c	3
d	4
ab	5
bc	6
ca	7
abc	8
cab	9
bcd	10

The program reads one character at a time. If the code is in the dictionary, then it adds the character to the current work string, and waits for the next one. This occurs on the first character as well. If the work string is not in the dictionary, (such as when the second character comes across), it adds the work string to the dictionary and sends over the wire (or writes to a file) the code assigned to the work string without the new character. It then sets the work string to the new character.

LZW Decompression

```
read fixed length token k (code or char)
output k
w = k
loop
    read a fixed length token k
    entry = dictionary entry for k
    output entry
    add w + first char of entry to
        the dictionary
    w = entry
endloop
```

The nice thing is that the decompressor builds its own dictionary on its side, that matches exactly the compressor's, so that only the codes need to be sent.