Dictionary Techniques

C.M. Liu
Perceptual Signal Processing Lab
College of Computer Science
National Chiao-Tung University

http://www.csie.nctu.edu.tw/~cmliu/Courses/Compression/



Office: EC538
(03)5731877
cmliu@cs.nctu.edu.tw

Contents

- Overview
- Introduction
- Static Dictionary
- Adaptive Dictionary
- Applications
- Summary

Overview

- **Statistical compression methods**
  - Use a statistical model of the data, and the quality of compression they achieve depends on how good that model is.
- **Dictionary-based compression methods**
  - Do not use a statistical model, nor do they use variable-size codes.
  - The dictionary holds strings of symbols and it may be static or dynamic (adaptive). The former is permanent, sometimes allowing the addition of strings but no deletions, whereas the latter holds strings previously found in the input stream, allowing for additions and deletions of strings as new input is being read.

PSP LAB

Introduction

- **So far we (largely) assumed symbol independence**
  - Not true for many common data types
    - E.g., text, images, code
- **Basic idea**
  - Identify frequent symbol patterns.
  - Encode those more efficiently.
  - Use a default (less efficient) encoding for the rest.
  - Hope to come out ahead.
- **Notes**
  - This looks reasonable for things like text.
  - Obviously won't work for (close to) random data.

Motivating Example

- **Consider an 'English' source with**
  - 26 letters & six punctuation marks.
- **Single-symbol, fixed-length encoding**
  - 5 bits per symbol.
- **Four-symbol patterns, fixed-length encoding**
  - 20 bits per symbol ($32^4 = 2^{20} = 1{,}048{,}576$).
- **Assume uneven distribution**
  - pick a ***dictionary*** the 256 most-frequent patterns and encode them w/ 8bits
  - encode the rest with 20 bits
  - must add 1-bit prefix to each to distinguish the two cases

Motivating Example (2)

- **If *p* is the probability of seeing a dictionary, then the bit rate is**

  $R = 9p + 21(1-p) = 21 - 12p$

- **Compression <=> 21 - 12p < 20**

  => $p > 0.084$

  - Not too bad
    - under equal probability assumption, $p = 0.00025$
- **Bottom line**
  - For a specific application, we could devise an appropriate dictionary to maximize compression

Digram Coding

- **Dictionary**
  - All letters from the alphabet +
  - As many digrams (pairs of letters) as possible
- **Example**
  - Dictionary size = 256 entries
  - Alphabet: printable ASCII = 95
  - Digrams: 161 most common pairs
- **Another example: $\mathcal{A}$ = {*a*, *b*, *c*, *d*, *r*},**
  - Dictionary:

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

PSP LAB

Digram Example

Input: abracadabra

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

Output: 101

Digram Example

Input: --racadabra

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

X

Output: 101

Digram Example

Input: --racadabra

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

Output: 101100

Digram Example

Input: ---acadabra

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

Output: 101100110

Digram Example

Input: -----adadabra

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

Output: 101100110111

Digram Example

Input: -------abra

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

Output: 101100110111101

Digram Example

Input： ---------ra

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

x

Output： 1011001101111101

Digram Example

Input： ---------ra

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

Output： 101100110111101100

Digram Example

Input： ----------a

Code	Entry	Code	Entry
000	*a*	100	*r*
001	*b*	101	*ab*
010	*c*	110	*ac*
011	*d*	111	*ad*

Output： 10110011011110110000

Problem: Which Digrams?

Pair	Count	Pair	Count
e␣	1128	ar	314
␣t	838	at	313
␣␣	823	␣w	309
th	817	te	296
he	712	␣s	295
in	512	d␣	272
s␣	494	␣o	266
er	433	io	257
␣a	425	co	256
t␣	401	re	247
en	392	␣$	246
on	385	r␣	239
n␣	353	di	230
ti	322	ic	229
␣i	317	ct	226

Source #1: Textbook chapter (LaTeX)

Pair	Count	Pair	Count
␣␣	5728	st	442
nl␣	1471	le	440
;nl	1133	ut	440
in	985	f(	416
nt	739	ar	381
=␣	687	or	374
␣i	662	r␣	373
t␣	615	en	371
␣=	612	er	358
);	558	ri	357
,␣	554	at	352
nlnl	506	pr	351
␣f	505	te	349
e␣	500	an	348
␣*	444	lo	347

Source #2: C code

PSP LAB

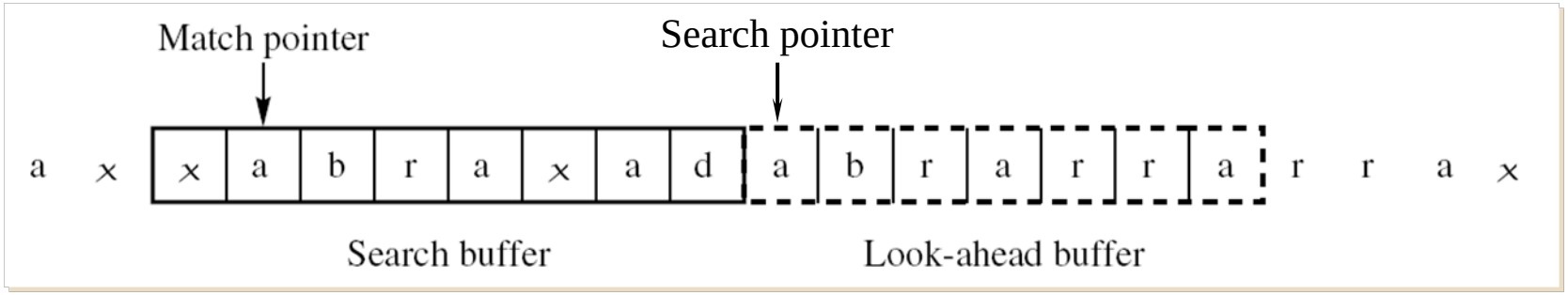
Building Dictionaries Adaptively

- **The beginning**
  - Jacob Ziv/Abraham Lempel
  - 1977 (LZ77/LZ1), 1978 (LZ78/LZ2)
  - 1984 Terry Welch (LZW)
- **LZ77 vs. LZ78**
  - Two different approaches
  - Multiple variation
  - Source of several mainstream compression schemes

LZ77

- **General approach**
  - Dictionary is a portion of the previously encoded sequence.
  - **sliding window compression**
- **Mechanism**
  - Search buffer, look-ahead buffer, search pointer
  - Goal:
    - find the maximum length match for the string pointed by the search pointer in the search buffer
    - encode it
  - Rationale
    - if patterns tend to repeat locally, we should get more efficient representation

LZ77 (2)



- **(o)ffset = search ptr - match ptr = 7**
- **(l)ength = number of consecutive letters matched = 4**
- **(c)odeword(r)**
- **Encoding**
  - <o, l, c> = <7, 4, C('r')>
- **If |search buff| = S, $|\mathcal{A}|$ = A, S + |LA buff| = W**
  - Fixed-length encoding: $\lceil \log_2 S \rceil + \lceil \log_2 W \rceil + \lceil \log_2 A \rceil$

LZ77 Encoding Example

- **Sequence**
  - cabracadabrarrarrad
  - W = 13, S = 7
- |cabraca|dabrar|rarrad
  - no match for **d**
  - send <0, 0, C(d)>
- |abracad|abrarr|arrad
  |abracad|abrarr|arrad
  |abracad|abrarr|arrad
  |abracad|abrarr|arrad
  - Send <7, 4, C(r)>

PSP LAB

LZ77 Encoding Example(2)

- |cadabrar|rarrad|
  |cadabrar|rarrad|
  |cadabrar|rarrad|
  - Send <3, 3, C(r)>
- **Could we do better?**
  - Send <3, 5, C(d)> instead!
- **Summary: three cases**
  - No match
  - Match
  - Match beyond the end of search buffer

LZ77 Decoding Example

- **Input**
  - <0, 0, C(d)> <7, 4, C(r)> <3, 5, C(d)>
- **Output**
  - cabraca
- **Decode: <0, 0, C(d)>**
  - Add a 'd': c|abracad|
- **Decode: <7, 4, C(r)>**
  - Start with the first 'a', copy four letters, decode C(r)
    cabrac|adabrar|
- **Decode: <3, 5, C(d)>**
  - Start with the first 'r', copy 3 letters:
    cabracada|brarrar|
  - copy 2 more letters
    cabracadabr|arrarar|
  - Decode C(d): cabracadabrarrararad

LZ77 Variants

- **So far**
  - Adaptive scheme, no prior knowledge
  - **Asymptotically** approaches case of full knowledge of source statistics
  - However, locality seems to play a non-trivial role
- **Impovements**
  - Variable-bit encoding
    - PKZip, Zip, Lharcm ONG, gzip, ARJ
  - Variable buffer size
    - Larger bugger requires faster searches
  - Elimination of <0, 0, C(x)>
    - LZSS: simply use a flag bit

LZ78

- **LZ77 assumes locality of patterns**
- **LZ78:**
  - No search buffer--explicit dictionary instead
  - Encoder/decoder must build dictionary in sync
  - Encoding: <i, c>
    - i = index in dictionary table
    - c = code of following character
- **Example:**
  - wabba□wabba□wabba□wabba□woo□woo□woo

LZ78 Example

Input: wabba□wabba□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry

Output:

LZ78 Example (2)

Input: wabba□wabba□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w

Output:

<0, C(w)>

LZ78 Example (3)

Input: -abba□wabba□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	**a**

Output:

<0, C(w)>
<0, C(a)>

LZ78 Example (4)

Input: --bba□wabba□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	**b**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>

LZ78 Example (5)

Input: ---ba□wabba□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	**ba**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>

LZ78 Example (6)

Input: -----□wabba□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	__

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>

LZ78 Example (7)

Input: ------**wa**bba□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	**wa**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>

LZ78 Example (8)

Input: --------bbaᐧwabbaᐧwabbaᐧwooᐧwooᐧwoo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	**bb**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>

LZ78 Example (9)

Input: ----------a□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	**a**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>

LZ78 Example (10)

Input: ----------wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	**wab**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>

LZ78 Example (11)

Input: ----------ba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	wab
10	**ba**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>
<4, C( )>

LZ78 Example (12)

Input: ----------wabbawoowoowoo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	wab
10	ba

11	wabb

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>
<4, C( )>
<9, C(b)>

LZ78 Example (13)

Input: ----------a□woo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	wab
10	ba

11	wabb
12	**a w**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>
<4, C( )>
<9, C(b)>
<8, C(w)>

LZ78 Example (14)

Input: ----------oo□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	wab
10	ba
11	wabb
12	a w
13	**o**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>
<4, C( )>
<9, C(b)>
<8, C(w)>
<0, C(o)>

PSP LAB

LZ78 Example (15)

Input: ----------o□woo□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	wab
10	ba
11	wabb
12	a w
13	o
14	**o**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>
<4, C( )>
<9, C(b)>
<8, C(w)>
<0, C(o)>
<13, C( )>

PSP LAB

LZ78 Example (16)

Input: ----------**wo**o□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	wab
10	ba
11	wabb
12	a w
13	o
14	o
15	**wo**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>
<4, C( )>
<9, C(b)>
<8, C(w)>
<0, C(o)>
<13, C( )>
<1, C(o)>

PSP LAB

LZ78 Example (17)

Input: ----------o□woo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	wab
10	ba
11	wabb
12	a w
13	o
14	o
15	wo
16	**o w**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>
<4, C( )>
<9, C(b)>
<8, C(w)>
<0, C(o)>
<13, C( )>
<1, C(o)>
<14, C(w)>

PSP LAB

LZ78 Example (18)

Input: ----------oo

Dictionary:

Index	Entry
1	w
2	a
3	b
4	ba
5	
6	wa
7	bb
8	a
9	wab
10	ba

11	wabb
12	a w
13	o
14	o
15	wo
16	o w
17	**oo**

Output:

<0, C(w)>
<0, C(a)>
<0, C(b)>
<3, C(a)>
<0, C( )>
<1, C(a)>
<3, C(b)>
<2, C( )>
<6, C(b)>
<4, C( )>
<9, C(b)>
<8, C(w)>
<0, C(o)>
<13, C( )>
<1, C(o)>
<14, C(w)>
<13, C(o)>

PSP LAB

LZ78 Notes

- **Observation**
  - If we keep on encoding the dictionary will keep on growing
- **Practical choices**
  - Stop growing the dictionary
    - effectively switch to a static dictionary
  - Prune it
    - e.g. based on usage statistics
  - Reset it
    - Start all over again
  - Without specific knowledge of the source, neither of these may work well!

LZ78 Variants: LZW

- **Terry Welch (1984)**
- **Idea**
  - Instead of <i, c>, encode i only
- **Algorithm**
  - Seed dictionary with all alphabet letters, p = null

```
while( !done)
  a = get_next_symbol
  if( p · a) is in dictionary
        p = p · a
  else
        send out index of p
        p = a
endwhile
```

LZW Encoding

Input: wabba□wabba□wabba□wabba□woo□woo□woo

Dictionary: p = Output:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	
7	
8	
9	
10	

LZW Encoding (2)

Input: wabba□wabba□wabba□wabba□woo□woo□woo

Dictionary: p = w Output:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	
7	
8	
9	
10	

LZW Encoding (3)

Input: -abba□wabba□wabba□wabba□woo□woo□woo

Dictionary:

p = wa

Output:

5 ('w')

Index	Entry
1	
2	a
3	b
4	o
5	w
6	**wa**
7	
8	
9	
10	

LZW Encoding (4)

Input: --bba□wabba□wabba□wabba□woo□woo□woo

Dictionary:

p = ab

Output:

5 (‘w’)
2 (‘a’)

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	**ab**
8	
9	
10	

LZW Encoding (5)

Input: ---ba□wabba□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	**bb**
9	
10	

p = bb

Output:

5 (‘w’)
2 (‘a’)
3 (‘b’)

LZW Encoding (6)

Input: ----a□wabba□wabba□wabba□woo□woo□woo

Dictionary:

p = ba

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	**ba**
10	

LZW Encoding (7)

Input: -----□wabba□wabba□wabba□woo□woo□woo

Dictionary:

p = a

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	**a**

LZW Encoding (8)

Input: ------wabba□wabba□wabba□woo□woo□woo

Dictionary: p = w

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	
13	
14	
15	
16	
17	
18	
19	
20	

LZW Encoding (9)

Input: -------abba□wabba□wabba□woo□woo□woo

Dictionary: p = wa

Index	Entry	Index	Entry
1		11	w
2	a	12	
3	b	13	
4	o	14	
5	w	15	
6	wa	16	
7	ab	17	
8	bb	18	
9	ba	19	
10	a	20	

Output:

5 (‘w’)
2 (‘a’)
3 (‘b’)
3 (‘b’)
2 (‘a’)
1 (‘ ’)

LZW Encoding (10)

Input: --------bba□wabba□wabba□woo□woo□woo

Dictionary:

p = wab

Output:

5 (‘w’)
2 (‘a’)
3 (‘b’)
3 (‘b’)
2 (‘a’)
1 (‘ ’)
6 (‘wa’)

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	**wab**
13	
14	
15	
16	
17	
18	
19	
20	

LZW Encoding (11)

Input: ---------ba□wabba□wabba□woo□woo□woo

Dictionary: p = bb

Output:

5 (‘w’)
2 (‘a’)
3 (‘b’)
3 (‘b’)
2 (‘a’)
1 (‘ ’)
6 (‘wa’)

Index	Entry	Index	Entry
1		11	w
2	a	12	wab
3	b	13	
4	o	14	
5	w	15	
6	wa	16	
7	ab	17	
8	bb	18	
9	ba	19	
10	a	20	

LZW Encoding (12)

Input: ----------a□wabba□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	**bba**
14	
15	
16	
17	
18	
19	
20	

p = bba

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')

LZW Encoding (13)

Input: ----------□wabba□wabba□woo□woo□woo

Dictionary:

p = a

Output:

5 (‘w’)
2 (‘a’)
3 (‘b’)
3 (‘b’)
2 (‘a’)
1 (‘ ’)
6 (‘wa’)
8 (‘bb’)

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Encoding (14)

Input: ----------wabba□wabba□woo□woo□woo

Dictionary: p = a w

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	**a w**
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Encoding (15)

Input: ----------abba□wabba□woo□woo□woo

Dictionary: p = wa

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	
16	
17	
18	
19	
20	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')

LZW Encoding (16)

Input: ----------bba□wabba□woo□woo□woo

Dictionary: p = wab

Output:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	
16	
17	
18	
19	
20	

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')

LZW Encoding (17)

Input: ----------ba□wabba□woo□woo□woo

Dictionary:

p = wabb

Output:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	**wabb**
16	
17	
18	
19	
20	

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')

LZW Encoding (18)

Input: ----------------a□wabba□woo□woo□woo

Dictionary:

p = ba

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	
17	
18	
19	
20	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')

LZW Encoding (19)

Input: ----------□wabba□woo□woo□woo

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	**ba**
17	
18	
19	
20	

p = ba

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')

PSP LAB

LZW Encoding (20)

Input: ----------wabba□woo□woo□woo

Dictionary: p = w

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	
18	
19	
20	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')

PSP LAB

LZW Encoding (21)

Input: ----------abba□woo□woo□woo

Dictionary: p = wa

Output:

Index	Entry	Index	Entry
1		11	w
2	a	12	wab
3	b	13	bba
4	o	14	a w
5	w	15	wabb
6	wa	16	ba
7	ab	17	**wa**
8	bb	18	
9	ba	19	
10	a	20	

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')

PSP LAB

LZW Encoding (22)

Input: ----------bba□woo□woo□woo

Dictionary: p = ab

Index	Entry	Index	Entry
1		11	w
2	a	12	wab
3	b	13	bba
4	o	14	a w
5	w	15	wabb
6	wa	16	ba
7	ab	17	wa
8	bb	18	
9	ba	19	
10	a	20	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')

PSP LAB

LZW Encoding (23)

Input: ----------ba□woo□woo□woo

Dictionary: p = abb

Index	Entry	Index	Entry
1		11	w
2	a	12	wab
3	b	13	bba
4	o	14	a w
5	w	15	wabb
6	wa	16	ba
7	ab	17	wa
8	bb	18	**abb**
9	ba	19	
10	a	20	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')

PSP LAB

LZW Encoding (24)

Input: ----------a□woo□woo□woo

Dictionary: p = ba

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	
20	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')

PSP LAB

LZW Encoding (25)

Input: ----------□woo□woo□woo

Dictionary:

p = ba

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	
20	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')

PSP LAB

LZW Encoding (26)

Input: ----------**w**oo□woo□woo

Dictionary: p = ba w

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	**ba w**
20	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

PSP LAB

LZW Encoding (27)

Input: ----------wo□woo□woo

Dictionary: p = wo

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	**wo**

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')

PSP LAB

LZW Encoding (28)

Input: ----------o□woo□woo

Dictionary: p = oo

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	**oo**
22	
23	
24	
25	
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')

PSP LAB

LZW Encoding (29)

Input: ----------□woo□woo

Dictionary: p = o

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	
24	
25	
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')

PSP LAB

LZW Encoding (30)

Input: ----------woo□woo

Dictionary: p = w

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	
24	
25	
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')

PSP LAB

LZW Encoding (31)

Input: ----------oo□woo

Dictionary: p = wo

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	**wo**
24	
25	
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')
11 (' w')

PSP LAB

LZW Encoding (32)

Input: ----------o□woo

Dictionary: p = oo

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	wo
24	
25	
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')
11 (' w')

PSP LAB

LZW Encoding (33)

Input: ----------□woo

Dictionary: p = oo

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	wo
24	**oo**
25	
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')
11 (' w')
21 ('oo')

PSP LAB

LZW Encoding (34)

Input: ----------woo

Dictionary: p = w

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	wo
24	oo
25	
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')
11 (' w')
21 ('oo')

PSP LAB

LZW Encoding (35)

Input: ----------oo

Dictionary: p = wo

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	wo
24	oo
25	
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')
11 (' w')
21 ('oo')

PSP LAB

LZW Encoding (36)

Input: ----------o

Dictionary: p = woo

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	wo
24	oo
25	wo o
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')
11 (' w')
21 ('oo')
23 (' wo')

PSP LAB

LZW Encoding (EOT)

Input: ----------

Dictionary: p = o

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	w
12	wab
13	bba
14	a w
15	wabb
16	ba
17	wa
18	abb
19	ba w
20	wo

Index	Entry
21	oo
22	o
23	wo
24	oo
25	wo o
26	
27	
28	
29	
30	

Output:

5 ('w')
2 ('a')
3 ('b')
3 ('b')
2 ('a')
1 (' ')
6 ('wa')
8 ('bb')
10 ('a ')
12 ('wab')
9 ('ba')
11 (' w')
7 ('ab')
16 ('ba ')

5 ('w')
4 ('o')
4 ('o')
11 (' w')
21 ('oo')
23 (' wo')
4 ('o')

PSP LAB

LZW Decoding

Input: 5 2 3 3 2 1 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output:

p =

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	
7	
8	
9	
10	

PSP LAB

LZW Decoding (2)

Input: 5 2 3 3 2 1 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: w

p = w

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	
7	
8	
9	
10	

PSP LAB

LZW Decoding (3)

Input: - 2 3 3 2 1 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wa

p = wa

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	**wa**
7	
8	
9	
10	

PSP LAB

LZW Decoding (4)

Input: - - 3 3 2 1 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wab

p = ab

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	**ab**
8	
9	
10	

PSP LAB

LZW Decoding (5)

Input: - - - 3 2 1 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabb

p = bb

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	**bb**
9	
10	

PSP LAB

LZW Decoding (6)

Input: - - - - 2 1 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba

p = ba

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	**ba**
10	

PSP LAB

LZW Decoding (7)

Input: - - - - - 1 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba␣

p = a

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	**a**␣

PSP LAB

LZW Decoding (8)

Input: - - - - - - - 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wa

p = wa

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	
13	
14	
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (9)

Input: - - - - - - - - 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wa

p = wa

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	
13	
14	
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (10)

Input: - - - - - - - - 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabb

p = wabb

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	**wab**
13	
14	
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (11)

Input: - - - - - - - - - 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabb

p = bb

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	**wab**
13	
14	
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (12)

Input: - - - - - - - - - 10 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba

p = bba

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	**bba**
14	
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (13)

Input: - - - - - - - - - - 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba

p = bba

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	**bba**
14	
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (14)

Input: - - - - - - - - - - 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba

p = a

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (15)

Input: - - - - - - - - - - 12 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba wab

p = a wab

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	**a w**
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (16)

Input: - - - - - - - - - - - - 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba wab

p = wab

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	a w
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (16)

Input: - - - - - - - - - - - 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba wab

p = wab

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	a w
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (17)

Input: - - - - - - - - - - - 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba wab

p = wab

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	a w
15	
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (18)

Input: - - - - - - - - - - - 9 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba wabba

p = wabba

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	a w
15	**wabb**
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (19)

Input: - - - - - - - - - - - - - 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba wabba

p = ba

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	a w
15	wabb
16	
17	
18	
19	
20	

PSP LAB

LZW Decoding (20)

Input: - - - - - - - - - - - - - 11 7 16 5 4 4 11 21 23 4

Output: wabba wabba wabba w

p = ba w

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	a w
15	wabb
16	**ba**
17	
18	
19	
20	

PSP LAB

LZW Decoding (21)

Input: - - - - - - - - - - - - - - - 7 16 5 4 4 11 21 23 4

Output: wabba wabba wabba w

p = w

Dictionary:

Index	Entry
1	
2	a
3	b
4	o
5	w
6	wa
7	ab
8	bb
9	ba
10	a

Index	Entry
11	**w**
12	wab
13	bba
14	a w
15	wabb
16	ba
17	
18	
19	
20	

… and so on …

PSP LAB

LZW Special Case

- **The presented decoder is**
  - simple, elegant, and … wrong
  - but fixable
- **Example:**
  - $\mathcal{A} = \{a, b\}$
  - Input: *abababab* …
- **Starting dictionary**

Index	Entry
1	a
2	b

LZW Special Case: Encoding

Input: abababababab …

Dictionary: p = Output:

Index	Entry
1	a
2	b

LZW Special Case: Encoding (2)

Input: ababababababab …

Dictionary: p = a Output:

Index	Entry
1	a
2	b

LZW Special Case: Encoding (3)

Input: -bababababab …

Dictionary:

Index	Entry
1	a
2	b
3	**ab**

p = ab

Output:

1 (‘a’)

LZW Special Case: Encoding (4)

Input: --ababababab …

Dictionary:

p = ba

Output:

Index	Entry
1	a
2	b
3	ab
4	**ba**

1 (‘a’)
2 (‘b’)

LZW Special Case: Encoding (5)

Input: ---bababababab …

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba

p = ab

Output:

1 (‘a’)
2 (‘b’)

LZW Special Case: Encoding (6)

Input: ----ababababab …

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba
5	**aba**

p = aba

Output:

1 (‘a’)
2 (‘b’)
3 (‘ab’)

LZW Special Case: Encoding (7)

Input: -----bababab …

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba
5	aba

p = ab

Output:

1 (‘a’)
2 (‘b’)
3 (‘ab’)

LZW Special Case: Encoding (8)

Input: ------ababab …

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba
5	aba

p = aba

Output:

1 (‘a’)
2 (‘b’)
3 (‘ab’)

LZW Special Case: Encoding (9)

Input: -------ababab …

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba
5	aba
6	abab

p = abab

Output:

1 (‘a’)
2 (‘b’)
3 (‘ab’)
5 (‘aba’)

LZW Special Case: Decoding

Input: 1 2 3 5 …

Output:

p =

Dictionary:

Index	Entry
1	a
2	b

We need to know in advance the initial dictionary used, but we can reconstruct the additional entries as we go, since they are always simply concatenations of previous entries.

the newly-created dictionary word is sent immediately

LZW Special Case: Decoding (2)

Input: 1 2 3 5 …

Output: a

p = a

Dictionary:

Index	Entry
1	a
2	b

LZW Special Case: Decoding (3)

Input: – 2 3 5 …

Output: ab

p = ab

Dictionary:

Index	Entry
1	a
2	b
3	**ab**

PSP LAB

LZW Special Case: Decoding (4)

Input: – – 3 5 …

Output: abab

p = bab

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba

LZW Special Case: Decoding (5)

Input: - - - 5 …

Output: abab

p = ab

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba

LZW Special Case: Decoding (6)

Input: - - - 5 …

Output: abab???

p = ab???

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba

LZW Special Case: Decoding (7)

Input: – – – 5 …

Output: abab???

p = ab???

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba
5	ab…

- **We know that 5th entry will start with 'ab'**

➔ **We can add it to *p* and continue decoding**

LZW Special Case: Decoding (8)

Input: - - - 5 …

Output: abab???

p = abab?

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba
5	**aba**

*But, since it was sent immediately, it must also start with "whatever comes next", and so must end with the same symbol it starts with, namely **a**.*

LZW Special Case: Decoding (9)

Input: - - - 5 …

Output: abab???

p = ababa

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba
5	**aba**

LZW Special Case: Decoding (10)

Input: – – – – …

Output: abababa

p = aba

Dictionary:

Index	Entry
1	a
2	b
3	ab
4	ba
5	aba

➔ Decoder must have an exception handler for such cases.

General case, **cScSc, where c is a single character, S is a string**

LZW Patents (http://en.wikipedia.org/wiki/Lempel-Ziv-Welch)

- Various patents have been issued in the United States and other countries for LZW and similar algorithms. LZ78 was covered by U.S. Patent 4,464,650 by Lempel, Ziv, Cohn, and Eastman, assigned to Sperry Corporation, later Unisys Corporation, filed on August 10, 1981. Two US patents were issued for the LZW algorithm: U.S. Patent 4,814,746 by Victor S. Miller and Mark N. Wegman and assigned to IBM, originally filed on June 1, 1983, and U.S. Patent 4,558,302 by Welch, assigned to Sperry Corporation, later Unisys Corporation, filed on June 20, 1983. On June 20, 2003, this patent on the LZW algorithm expired [1].
- US Patent 4,558,302 received a lot of negative press after LZW compression was used in the GIF image format (see Graphics Interchange Format#Unisys and LZW patent enforcement).

Applications: compress

- **An early implementation of LZW**
  - Adaptive dictionary
  - Start with 512 entries (9-bit codewords)
  - When it fills up— doubles dictionary size (codewords become longer by a bit)
  - Max codeword length b_{max} = 9..16
  - When dictionary reaches $2^{b_{max}}$ entries
    - it becomes a static dictionary encoder
    - If compression ration falls below a threshold, dictionary is reset

Applications: GIF



- **LZW scheme, similar to compress:**
  - Clear code: ***b*** bits/pixel → 2^b clear codes
  - Starting dictionary → 2^{b+1}
  - Dictionary size doubles *as in* **compress**
  - Max size = 4096 entries, static dictionary
  - Format:
    - Codewords stored in blocks of characters up to 255
    - Each block has a header with count
    - Block terminator: 8 zero bits
    - End of image info: 2^b+1 (before block terminator)

PSP LAB

GIF Performance

Image	GIF	Arithmetic Coding of Pixel Values	Arithmetic Coding of Pixel Differences
Sena	51,085	53,431	31,847
Sensin	60,649	58,306	37,126
Earth	34,276	38,248	32,137
Omaha	61,580	56,061	51,393

Applications: PNG

- **Patent-free alternative to GIF (libpng.org)**
- **Designed specifically for lossless image compression**
- **Modes**
  - True color, grayscale, 8-bit pallette
- **Two autonomous compression components**
  - ***deflate*** (RFC 1951)—LZ77-style dictionary compression algorithm + Huffman
  - ***filtering***—lossless transformations of byte-level image data

Deflate

- **Deflate = LZ77 + Huffman**
- **Three types of (*zlib*) data blocks**
  - Uncompressed, LZ77 + fixed Huffman, LZ77 + dynamic Huffman
- **Match is b/w 3 & 258 bytes**
  - A sliding window of 3 bytes is examined
  - If match is not found, encode first byte and slide window`
- **LZ77 output**
  - literal
  - <match_length, offset>
- **Combined alphabet = literals + match_lengths**
  - Codes 0-255 == literals
  - 256 = end-of-block
  - 257-285: range
  - followed by *selector bits*

Deflate Match Length Codes

Index	# of selector bits	Length	Index	# of selector bits	Length	Index	# of selector bits	Length
257	0	3	267	1	15,16	277	4	67–82
258	0	4	268	1	17,18	278	4	83–98
259	0	5	269	2	19–22	279	4	99–114
260	0	6	270	2	23–26	280	4	115–130
261	0	7	271	2	27–30	281	5	131–162
262	0	8	272	2	31–34	282	5	163–194
263	0	9	273	3	35–42	283	5	195–226
264	0	10	274	3	43–50	284	5	227–257
265	1	11, 12	275	3	51–58	285	0	258
266	1	13, 14	276	3	59–66			

PSP LAB

Huffman Codes for Match Lengths

Index Ranges	# of bits	Binary Codes
0–143	8	00110000 through 10111111
144–255	9	110010000 through 111111111
256–279	7	0000000 through 0010111
280–287	8	11000000 through 11000111

- **Note: *offset* (1..32768) is coded with a different Huffman code!**

PSP LAB

Match Length Code

- 0–3: distances 1–4
- 4–5: distances 5–8, 1 extra bit
- 6–7: distances 9–16, 2 extra bits
- 8–9: distances 17–32, 3 extra bits
- ...
- 26–27: distances 8193–16384, 12 extra bits
- 28–29: distances 16385–32768, 13 extra bits
- 30–31: not used, reserved and illegal but still part of the tree.

PNG: Compression Filters

- **Image layout**
  - ***Scanlines*:** left-to-right, top-to-bottom
- **Filters are applied on a scanline-by-scanline basis**
- **All algorithms applied to bytes (not pixels!)**
- **Filter types**
  - None
    - Unmodified value
  - Sub
    - Difference from previous value (mod 256)
  - Up
    - Like Sub, except the previous pixel is above
  - Average
    - Subtract average(left, above)
  - Paeth
    - Difference with nearest(left, above, upper_left)

PNG: Performance

Image	PNG	GIF	Arithmetic Coding of Pixel Values	Arithmetic Coding of Pixel Differences
Sena	31,577	51,085	53,431	31,847
Sensin	34,488	60,649	58,306	37,126
Earth	26,995	34,276	38,248	32,137
Omaha	50,185	61,580	56,061	51,393

Summary

- **Introduced dictionary techniques**
  - Tend to work well for text & communication protocols
  - Static techniques are suitable for known data domain
- **Adaptive techniques**
  - LZ77
  - LZ78
  - LZW
- **Combination with Huffman**
  - Deflate
- **Practical applications**
  - GIF, zlip --> png, pdf, …

Homeworks

- **Homeworks (Sayood 3rd, pp.139-140)**
  - Deadline April 2
  - 3, 4, 5, 6, 7, 8