Lecture 3 - Compression

PORTLAND STATE UNIVERSITY

Administrative

- Programming assignment on web site.
  - Due Monday, October 19, 2015
  - Start early
- HW 1 – Due on Wednesday
- QUIZ #1 – Next Wednesday
  - October 14, 2015

PORTLAND STATE UNIVERSITY

Huffman encoding

Optimal encoding one symbol → one codeword

Huffman codebook required

- optimized for single instance
  - -) have to include in compressed file
- pick generic codebook (eg. English dictionary Huffman table)
  - +) no codebook in file
  - -) not quite as small

PORTLAND STATE UNIVERSITY

Run Length Encoding

Sequence of data + encode it as

<symbol, run length count>

- applicable to repeating data

1111 1 222 66

300 20 100

<1, 300> <2, 20> <6, 100>

Need to be careful w/ how to encode

- Effective for graphical images
- Limited use in text files
- JPEG uses this as part of its lossless encoding → entropy encoding process

PORTLAND STATE UNIVERSITY

Compression Fundamental

- CF3 – Escape sequences are necessary for control and disambiguation of data

AAAAA...A B.....B C.....C
9 12 7
A9B12C7

Create special characters or sequences that never occur in regular data
Have a special meaning
Example printf

printf("\"");
"

printf("\\\"");
\"

PORTLAND STATE UNIVERSITY

Run Length Encoding Variations

Variation 1
AABB CC DD EEEE FFFFF
A2B2C2D2E4F5
- wait for two symbols, then use RLE

AA1BB1CC1DD1EE3FF4

Variation 2
- Use escape sequence to indicate run

AB EEEE DDC BBBBBB *c = escape char
ABE*4D*2CB*6

Variation 3
- Using runs to skip a single common symbol

8 0002 4 005 000003
8([4],2)([1],4)([3],5)([7],3)
skipped places

JPEG uses this form

PORTLAND STATE UNIVERSITY

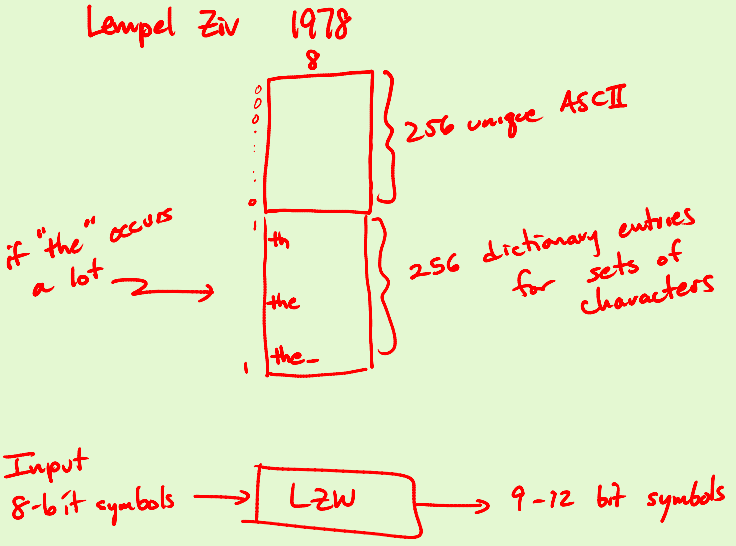
Adaptive Compression

- Huffman compression takes one symbol and generates variable length codeword
  - Decompression complicated because of variable bit processing
- Alternative: Learn about common sequences that occur in the input stream and generate one codeword

Common sequences in English, for example
the
and

PORTLAND STATE UNIVERSITY

LZW Compression



PORTLAND STATE UNIVERSITY

LZW Compression – summary
(accidentally skipped but was discussed)

- Lempel-Ziv Compression
  - convert variable length strings into fixed length codes
  - builds "table" during both compression and decompression
    - +) No table
    - -) Needs conditioning
- Lempel-Ziv-Welch (LZW) Compression - is essentially a fast implementation of the LZ compression method
  - Uses:
    - UNIX compress and gzip use LZ and LZW
    - v.42bis network data compression
    - gif and tiff image compression format

PORTLAND STATE UNIVERSITY

LZW Compression

LZW Compression

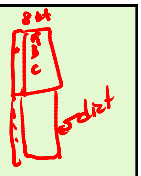
```
w = nil
while (read char k)
        if wk exists in dictionary
                w = wk
        else
                add wk to dictionary
                output code for w
                w = k
```

sequences that we have seen

sequences we haven't seen

PORTLAND STATE UNIVERSITY

Example: ABCDABCABBABBDABCE



w	k	dict		output
	A			
A	B	AB	<256>	A
B	C	BC	<257>	B
C	D	CD	<258>	C
D	A	DA	<259>	D
A	B			
AB	C	ABC	<260>	<256>
C	A	CA	<261>	C
A	B			
AB	B	ABB	<262>	<256>
B	A	BA	<263>	B
A	B			
AB	B			
ABB	D	ABBD	<264>	<262>

9-bit output symbols

ABC...D padded w/ 0's @ front

PORTLAND STATE UNIVERSITY

LZW Decompression

LZW Compression

Read *k*
Output *k*
w = *k*
while (read char *k*)
    entry = dictionary entry for *k*
    output *entry*
    add *w* + *entry[0]* to dictionary
    w = *entry*

compressed codeword

PORTLAND STATE UNIVERSITY

A B C <257> D <258> E <257>

W	k	k-dict	dictionary	output
A	A	B	AB <256>	A
B	B	C	BC <257>	B
C	<257>	BC	CB <258>	C
BC	D	D	BCD <259>	BC
D	<258>	CB	DC <260>	D
CB	E	E	CBE <261>	CB
E	<257>	BC	EB <262>	E

PORTLAND STATE UNIVERSITY

KwKwK String

K is a single symbol

w is a dictionary entry or single symbol

Kw exists in dictionary before KwKwK appears

w + entry[0] → dictionary

w + w[0] → dictionary

PORTLAND STATE UNIVERSITY