# Information Theory

Mohamed Hamada

Software Engineering Lab
The University of Aizu

Email: hamada@u-aizu.ac.jp
URL: http://www.u-aizu.ac.jp/~hamada

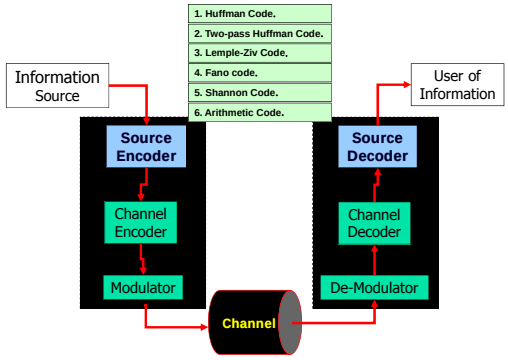


## Today's Topics

- Source Coding Techniques
- Huffman Code
- Two-pass Huffman Code
- Lemple-Ziv Encoding
- Lemple-Ziv Decoding



## Source Coding Techniques






## Source Coding Techniques

1. Huffman Code.
2. Two-pass Huffman Code.
3. Lemple-Ziv Code.
4. Fano code.
5. Shannon Code.
6. Arithmetic Code.



## Source Coding Techniques

1. Huffman Code.
2. Two-path Huffman Code.
3. Lemple-Ziv Code.
4. Fano Code.
5. Shannon Code .
6. Arithmetic Code.



## Source Coding Techniques

1. Huffman Code.

With the Huffman code in the binary case the two least probable source output symbols are joined together, resulting in a new message alphabet with one less symbol

1 take together smallest probabilites: P(i) + P(j)
2 replace symbol i and j by new symbol
3 go to 1 - until end

Application examples: JPEG, MPEG, MP3

1. Huffman Code.

**ADVANTAGES:**

- ***uniquely*** decodable code
- ***smallest*** average codeword length

**DISADVANTAGES:**

- LARGE tables give complexity
- sensitive to channel errors



1. Huffman Code.

For ***COMPUTER DATA*** data reduction is

***lossless*** → no errors at reproduction
***universal*** → effective for different types of data

Huffman is **not universal**!
it is only valid for one particular type of source:

**If the source has no probability distribution Huffman code can not applied.**



## Huffman Coding: Example

- Compute the Huffman Code for the source shown

| Source Symbol $s_k$ | Symbol Probability $p_k$ |
|---|---|
| $s_0$ | 0.1 |
| $s_1$ | 0.2 |
| $s_2$ | 0.4 |
| $s_3$ | 0.2 |
| $s_4$ | 0.1 |

**Note that: the entropy of S is**

$$H(S) = (0.4)\log_2\left(\frac{1}{0.4}\right) + 2\times(0.2)\log_2\left(\frac{1}{0.2}\right) + 2\times(0.1)\log_2\left(\frac{1}{0.1}\right) = 2.12193$$



## Solution A

| Source Symbol $s_k$ | Stage I |
|---|---|
| $s_2$ | 0.4 |
| $s_1$ | 0.2 |
| $s_3$ | 0.2 |
| $s_0$ | 0.1 |
| $s_4$ | 0.1 |



## Solution A

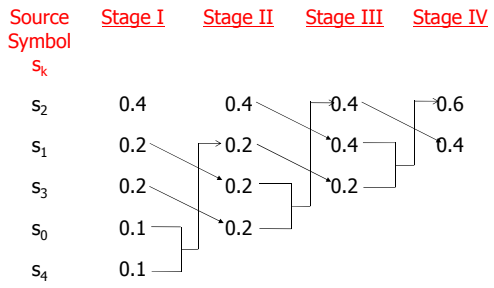
| Source Symbol $s_k$ | Stage I | Stage II |
|---|---|---|
| $s_2$ | 0.4 | 0.4 |
| $s_1$ | 0.2 | 0.2 |
| $s_3$ | 0.2 | 0.2 |
| $s_0$ | 0.1 | 0.2 |
| $s_4$ | 0.1 | |



## Solution A

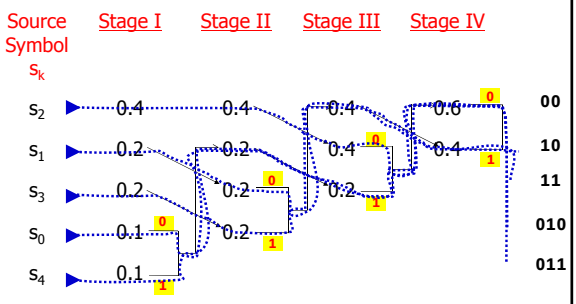
| Source Symbol $s_k$ | Stage I | Stage II | Stage III |
|---|---|---|---|
| $s_2$ | 0.4 | 0.4 | 0.4 |
| $s_1$ | 0.2 | 0.2 | 0.4 |
| $s_3$ | 0.2 | 0.2 | 0.2 |
| $s_0$ | 0.1 | 0.2 | |
| $s_4$ | 0.1 | | |

## Solution A



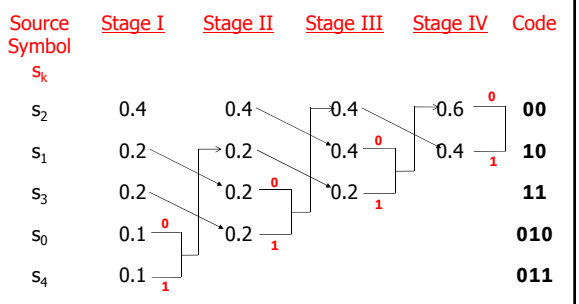

13

## Solution A




14

## Solution A




15

## Solution A Cont'd

| Source Symbol $s_k$ | Symbol Probability $p_k$ | Code word $c_k$ |
|---|---|---|
| $s_0$ | 0.1 | 010 |
| $s_1$ | 0.2 | 10 |
| $s_2$ | 0.4 | 00 |
| $s_3$ | 0.2 | 11 |
| $s_4$ | 0.1 | 011 |

$$H(S) = 2.12193$$

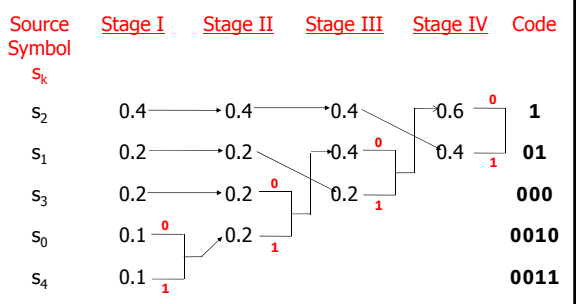
$$L = 0.4\times 2 + 0.2\times 2 + 0.2\times 2 + 0.1\times 3 + 0.1\times 3 = 2.2$$

$$H(S) \le L < H(S) + 1$$

**THIS IS NOT THE ONLY SOLUTION!**

16

## Another Solution B




17

## Another Solution B Cont'd

| Source Symbol $s_k$ | Symbol Probability $p_k$ | Code word $c_k$ |
|---|---|---|
| $s_0$ | 0.1 | 0010 |
| $s_1$ | 0.2 | 01 |
| $s_2$ | 0.4 | 1 |
| $s_3$ | 0.2 | 000 |
| $s_4$ | 0.1 | 0011 |

$$H(S) = 2.12193$$

$$L = 0.4\times 1 + 0.2\times 2 + 0.2\times 3 + 0.1\times 4 + 0.1\times 4 = 2.2$$

$$H(S) \le L < H(S) + 1$$

18

## What is the difference between the two solutions?

- They have the same average length
- They differ in the variance of the average code length

$$\sigma^2 = \sum_{k=0}^{K-1} p_k \left(l_k - L\right)^2$$

- Solution A
  - $\sigma^2$=0.16
- Solution B
  - $\sigma^2$=1.36

19

## Source Coding Techniques

1. Huffman Code.
2. **Two-pass Huffman Code.**
3. Lemple-Ziv Code.
4. Fano Code.
5. Shannon Code.
6. Arithmetic Code.

20

## Source Coding Techniques

**2. Two-pass Huffman Code.**

**This method is used when the probability of symbols in the information source is unknown. So we first can estimate this probability by calculating the number of occurrence of the symbols in the given message then we can find the possible Huffman codes. This can be summarized by the following two passes.**

**Pass 1 : Measure the occurrence possibility of each character in the message**

**Pass 2 : Make possible Huffman codes**

21

## Source Coding Techniques

**2. Two-pass Huffman Code.**

**Example**

**Consider the message:** M=ABABABABABACADABACADABACADABACAD

L(M)=32

#(A)=16 → p(A)=16/32=0.5
#(B)=8 → p(B)=8/32=0.25
#(C)=4 → p(C)=4/32=0.125
#(D)=4 → p(D)=4/32=0.125

| Symbol | Fraction | Pass 1 | Pass 2 | |
|---|---|---|---|---|
| | | Probability | Huffman (Comma) Codes | |
| A | 16/32 | 0.5 | 0 | 1 |
| B | 8/32 | 0.25 | 10 | 01 |
| C | 4/32 | 0.125 | 110 | 001 |
| D | 4/32 | 0.125 | 111 | 000 |

22

## Source Coding Techniques

1. Huffman Code.
2. Two-path Huffman Code.
3. **Lemple-Ziv Code.**
4. Fano Code.
5. Shannon Code.
6. Arithmetic Code.

23

## Lempel-Ziv Coding

- Huffman coding requires knowledge of a probabilistic model of the source
  - This is not necessarily always feasible
- Lempel-Ziv code is an adaptive coding technique that does not require prior knowledge of symbol probabilities
- Lempel-Ziv coding is the basis of well-known ZIP for data compression

24

## Lempel-Ziv Coding History

- **Universal:** effective for different types of data
- **Lossless:** no errors at reproduction

**Applications:**

GIF, TIFF, V.42bis modem compression standard, PostScript Level 2

**History:**

- 1977 published by Abraham Lempel and Jakob Ziv
- 1984 LZ-Welch algorithm published in IEEE Computer
  - Sperry patent transferred to Unisys (1986)
    GIF file format Required use of LZW algorithm

25

## Lempel-Ziv Coding Example

**Input:** 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1...

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | | | | | | | |
| Representation | | | | | | | | | |
| Encoding | | | | | | | | | |

26

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1...

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | | | | | | |
| Representation | | | | | | | | | |
| Encoding | | | | | | | | | |

27

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1...

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | | | | | |
| Representation | | | | | | | | | |
| Encoding | | | | | | | | | |

28

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1...

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | | | | |
| Representation | | | | | | | | | |
| Encoding | | | | | | | | | |

29

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1...

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | | | |
| Representation | | | | | | | | | |
| Encoding | | | | | | | | | |

30

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1...

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | 010 | | |
| Representation | | | | | | | | | |
| Encoding | | | | | | | | | |

31

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1...

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | 010 | 100 | |
| Representation | | | | | | | | | |
| Encoding | | | | | | | | | |

32

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1..

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | 010 | 100 | 101 |
| Representation | | | | | | | | | |
| Encoding | | | | | | | | | |

33

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1..

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | 010 | 100 | 101 |
| Representation | | | 1 | 2 | 2 | 1 | 1 | 1 | 2 |
| Encoding | | | | | | | | | |

34

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1..

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | 010 | 100 | 101 |
| Representation | | | 11 | 12 | 42 | 21 | 41 | 61 | 62 |
| Encoding | | | | | | | | | |

35

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1..

| Decimal | Binary |
|---|---|
| 1 | 001 |
| 2 | 010 |
| 4 | 100 |
| 6 | 110 |

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | 010 | 100 | 101 |
| Representation | | | 11 | 12 | 42 | 21 | 41 | 61 | 62 |
| Encoding | | | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

36

## Lempel-Ziv Coding Example

0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1 ...

| Decimal | Binary |
|---|---|
| 1 | 001 |
| 2 | 010 |
| 4 | 100 |
| 6 | 110 |

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | 010 | 100 | 101 |
| Representation | | | 11 | 12 | 42 | 21 | 41 | 61 | 62 |
| Encoding | | | 001 0 | 001 1 | 100 1 | 010 0 | 100 0 | 110 0 | 110 1 |

37

## Lempel-Ziv Coding Example

Information bits: 0 0 | 0 1 | 0 1 1 | 1 0 | 0 1 0 | 1 0 0 | 1 0 1 ...

Source encoded bits: 0010 | 0011 | 1001 | 0100 | 1000 | 1100 | 1101

| Codebook Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Subsequence | 0 | 1 | 00 | 01 | 011 | 10 | 010 | 100 | 101 |
| Representation | | | 11 | 12 | 42 | 21 | 41 | 61 | 62 |
| Source Code | | | 0010 | 0011 | 1001 | 0100 | 1000 | 1100 | 1101 |

38

## How Come this is Compression?!

- The hope is:
  - If the bit sequence is long enough, eventually the fixed length code words will be shorter than the length of subsequences they represent.
- When applied to English text
  - Lempel-Ziv achieves approximately 55%
  - Huffman coding achieves approximately 43%

39

## Encoding idea Lempel Ziv Welch-LZW

Assume we have just read a **segment** *w* from the text.
*a* is the next symbol.

*wa* | *a* | *w*

If *wa* is not in the dictionary,
Write the index of *w* in the **output file**.
Add *wa* to the dictionary, and set *w* ← *a*.

*wa* is in the dictionary,
**Process** the next symbol with segment *wa*.

40

## LZ Encoding example

Consider the input message:
Input string: a a b a a c a b c a b

**LZ encoding process:**

Dictionary
Initial
a 0
b 1
c 2

| Input: a a b a a c a b c a b | output | update |
|---|---|---|
| a a — aa not in dictionry, output 0 add aa to dictionary | 0 | aa 3 |
| a a b — continue with a, store ab in dictionary | 0 | ab 4 |
| a a b a — continue with b, store ba in dictionary | 1 | ba 5 |
| a a b a a c — aa in dictionary, aac not, | 3 | aac 6 |
| a a b a a c a | 2 | ca 7 |
| a a b a a c a b c | 4 | abc 8 |
| a a b a a c a b c a b | 7 | cab 9 |
| a a b a a c a b c a b | 1 | |

aabaacabcab → LZ Encoder → 00132471

41

## UNIVERSAL (LZW) (decoder)

1. Start with basic symbol set
2. Read a code *c* from the compressed file.
   - *The address c* in the dictionary determines the segment *w*.
   - write *w* in the output file.
3. Add *wa* to the dictionary: *a* is the first letter of the next segment

42

# LZ Decoding example

**Dictionary**

Initial

| | |
|---|---|
| a | 0 |
| b | 1 |
| c | 2 |

**Output String:**

| Input | update |
|---|---|
| **a** ? Output a | 0 | |
| **a a** ! output a determines ? = a, update aa | 0 | aa 3 |
| **a a b** . output 1 determines !=b, update ab | 1 | ab 4 |
| **a a b a a** . | 3 | ba 5 |
| **a a b a a c** . | 2 | aac 6 |
| **a a b a a c a b** . | 4 | ca 7 |
| **a a b a a c a b c a** . | 7 | abc 8 |
| **a a b a a c a b c a b** | 1 | |

00132471 → **LZ Decoder** → aabaacabcab

43

## Exercise

**1. Find Huffman code for the following source**

| Symbol | Probability |
|---|---|
| h | 0.1 |
| e | 0.1 |
| l | 0.4 |
| o | 0.25 |
| w | 0.05 |
| r | 0.05 |
| d | 0.05 |

**2. Find LZ code for the following input**

**0011001111010100010001001**

44