
Information Theory

Mohamed Hamada

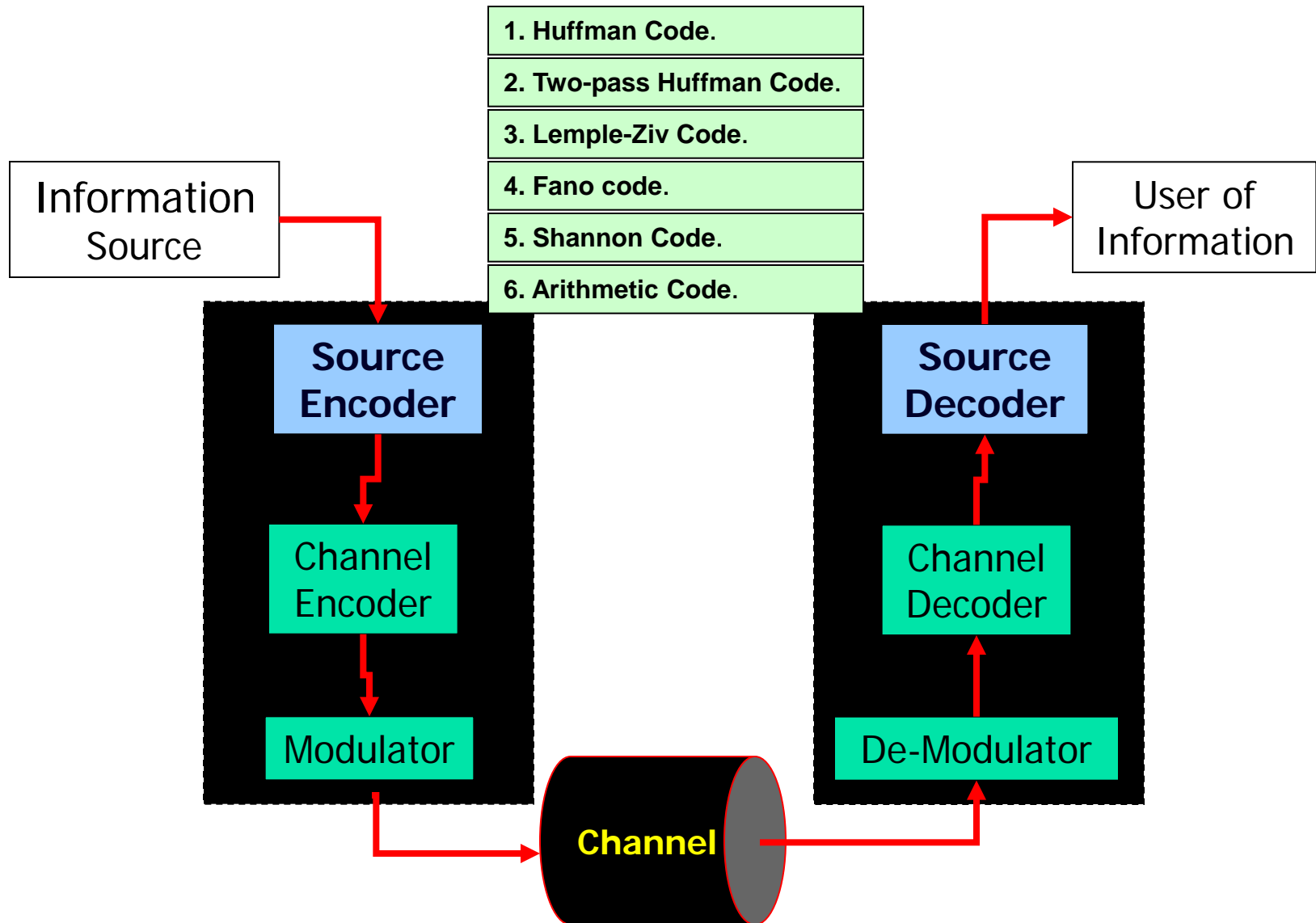
**Software Engineering Lab
The University of Aizu**

**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 probabilities: $P(i) + P(j)$
- 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

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

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

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

Solution A

Source
Symbol

Stage I

s_k

s_2 0.4

s_1 0.2

s_3 0.2

s_0 0.1

s_4 0.1

Solution A

Source
Symbol

Stage I

Stage II

s_k

s_2

0.4

0.4

s_1

0.2

0.2

s_3

0.2

0.2

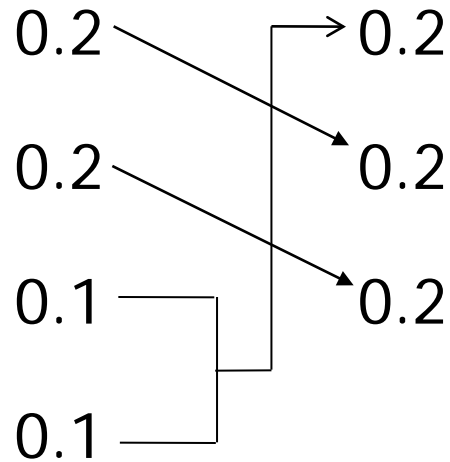
s_0

0.1

0.2

s_4

0.1



Solution A

Source
Symbol

Stage I

Stage II

Stage III

s_k

s_2

0.4

0.4

0.4

s_1

0.2

0.2

0.4

s_3

0.2

0.2

0.2

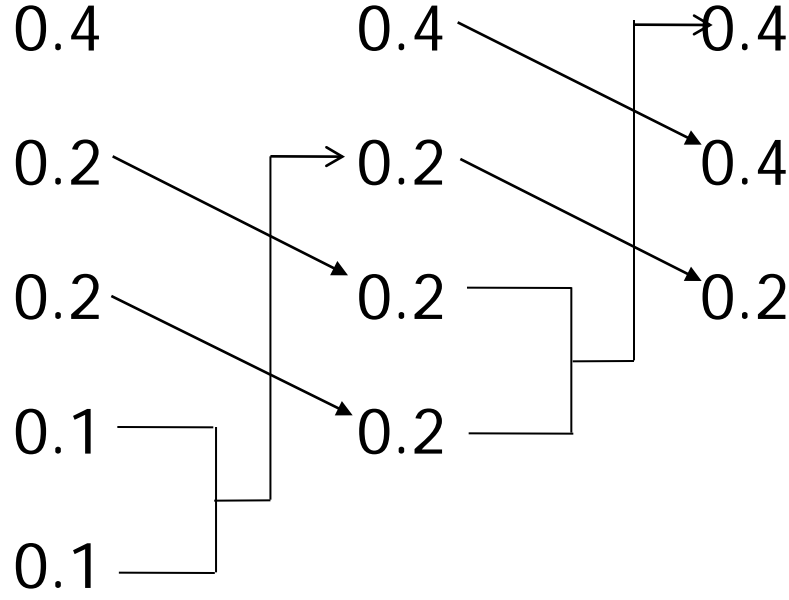
s_0

0.1

0.2

s_4

0.1



Solution A

Source
Symbol

Stage I

Stage II

Stage III

Stage IV

s_k

s_2

0.4

0.4

0.4

0.6

s_1

0.2

0.2

0.4

0.4

s_3

0.2

0.2

0.2

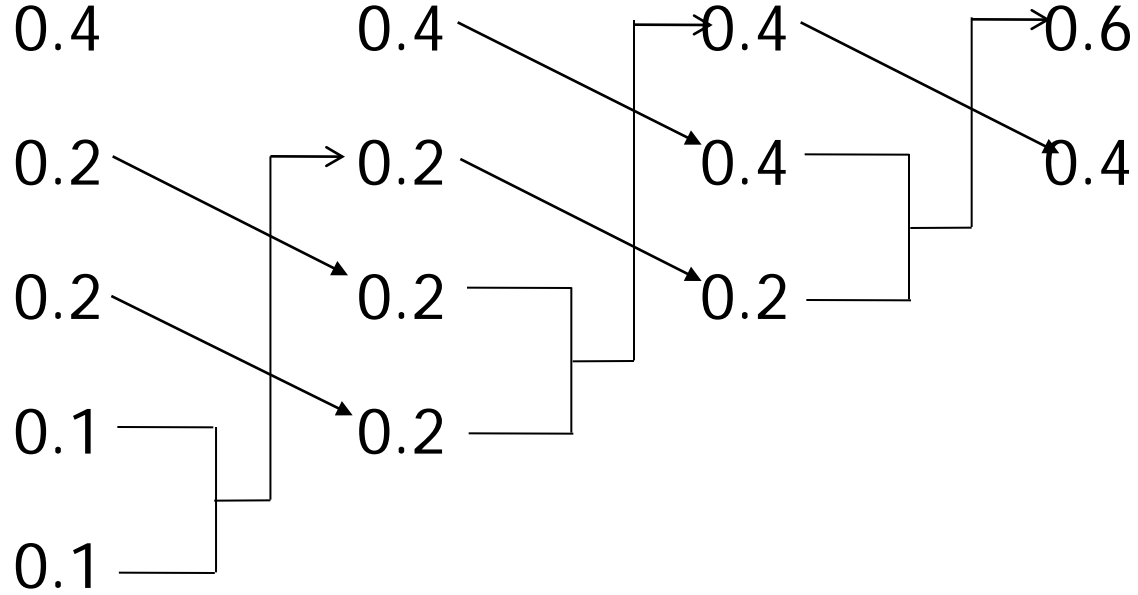
s_0

0.1

0.2

s_4

0.1



Solution A

Source
Symbol

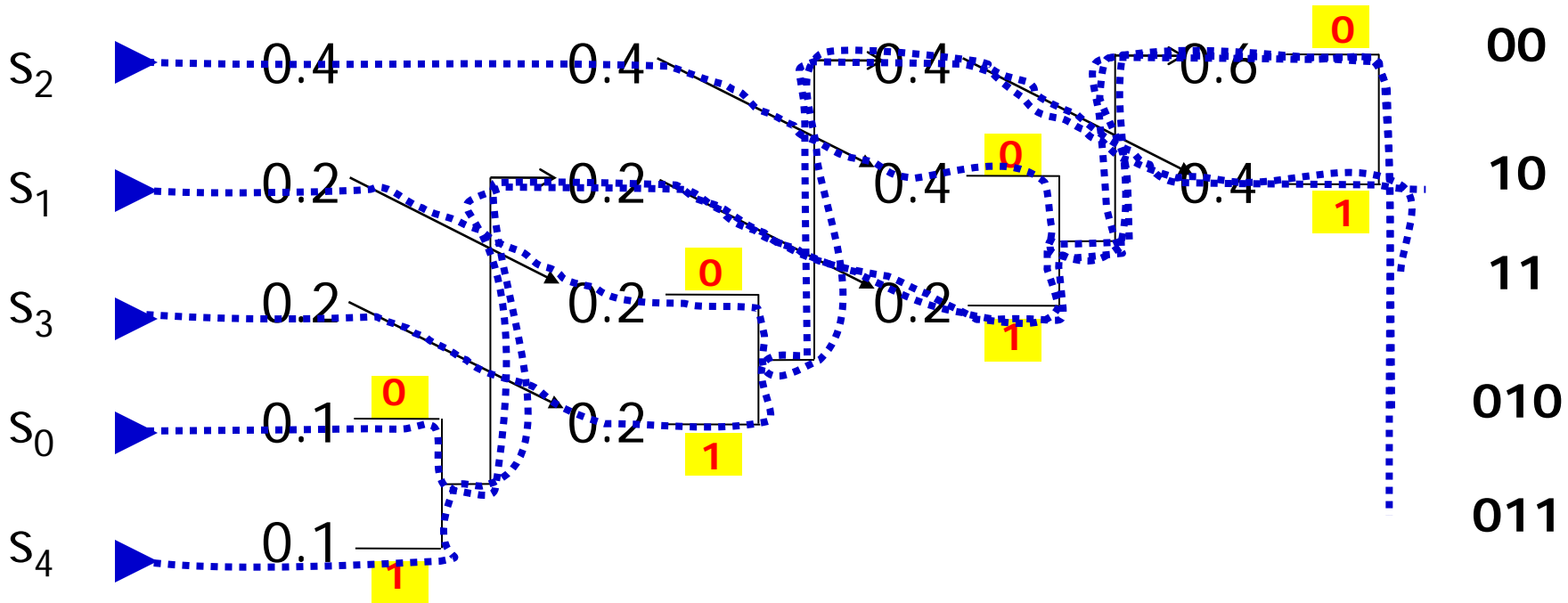
Stage I

Stage II

Stage III

Stage IV

s_k



Solution A

Source
Symbol

Stage I

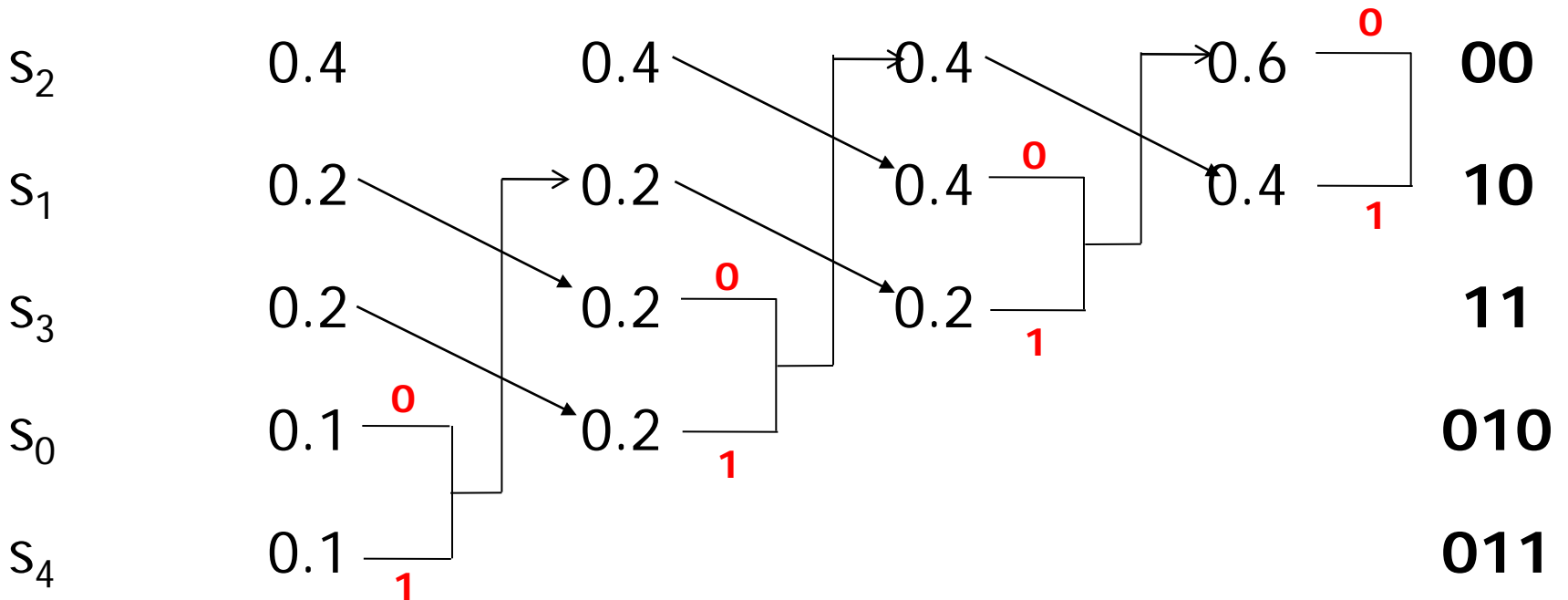
Stage II

Stage III

Stage IV

Code

s_k



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

$$H(S) = 2.12193$$

$$\begin{aligned} L &= 0.4 \times 2 + 0.2 \times 2 \\ &+ 0.2 \times 2 + 0.1 \times 3 + 0.1 \times 3 \\ &= 2.2 \end{aligned}$$

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

THIS IS NOT THE ONLY SOLUTION!

Another Solution B

Source
Symbol

Stage I

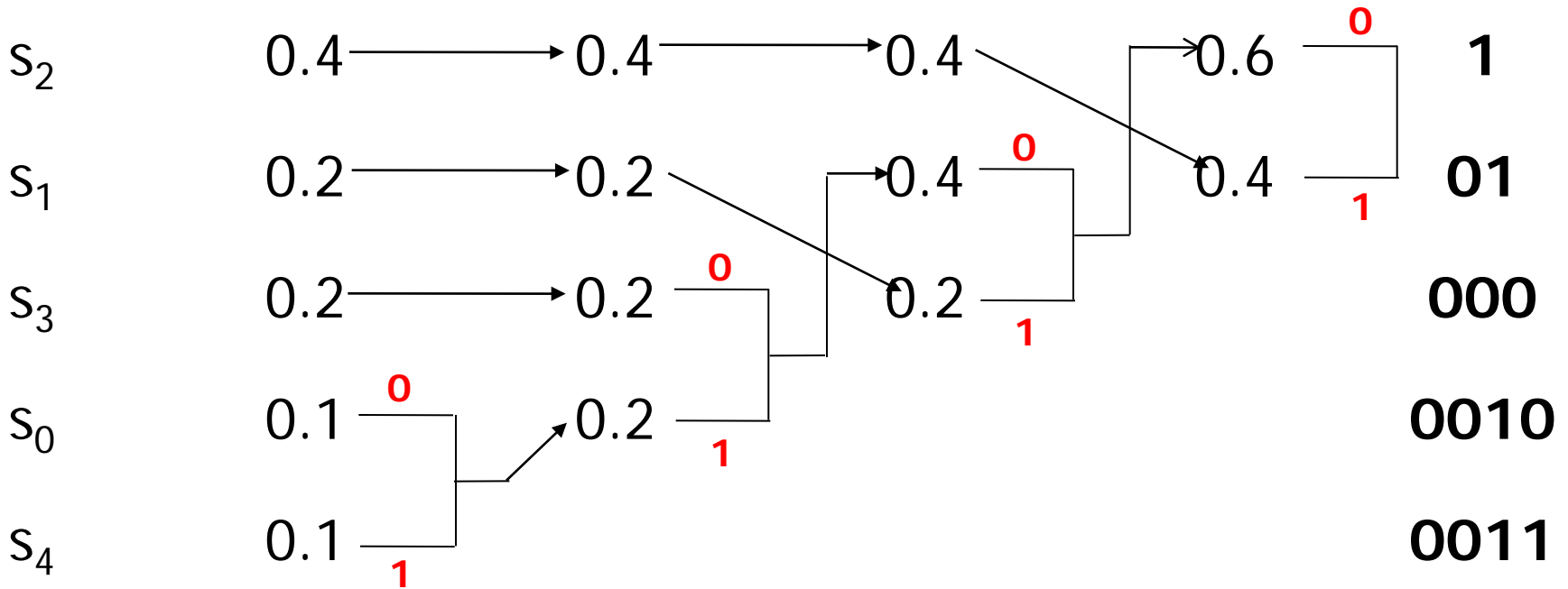
Stage II

Stage III

Stage IV

Code

s_k



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

$$H(S) = 2.12193$$

$$\begin{aligned} L &= 0.4 \times 1 + 0.2 \times 2 \\ &+ 0.2 \times 3 + 0.1 \times 4 + 0.1 \times 4 \\ &= 2.2 \end{aligned}$$

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

What is the difference between the two solutions?

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

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

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

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

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

Source Coding Techniques

2. Two-pass Huffman Code.

Example

Consider the message: $M=ABABABABABACADABACADABACADABACAD$

$$L(M) = 32$$

$$\#(A) = 16 \longrightarrow p(A) = 16/32 = 0.5$$

$$\#(B) = 8 \longrightarrow p(B) = 8/32 = 0.25$$

$$\#(C) = 4 \longrightarrow p(C) = 4/32 = 0.125$$

$$\#(D) = 4 \longrightarrow p(D) = 4/32 = 0.125$$

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

Source Coding Techniques

1. Huffman Code.

2. Two-path Huffman Code.

3. Lemple-Ziv Code.

4. Fano Code.

5. Shannon Code.

6. Arithmetic Code.

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

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

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									

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									

Lempel-Ziv Coding Example

000101110010100101...

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

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									

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									

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									

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									

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									

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									

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									

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

Lempel-Ziv Coding Example

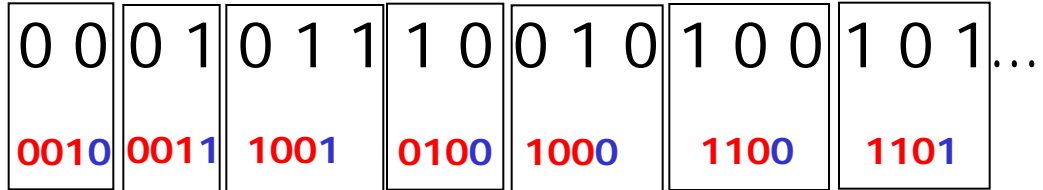
0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1 ..

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

Lempel-Ziv Coding Example

Information bits



Source encoded bits

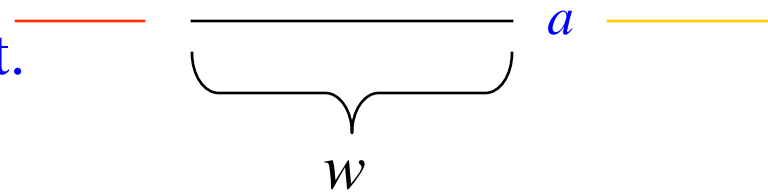
Codebook Index	1	2	3	4	5	6	7	8	9
Subsequence	0	1	00	01	011	10	010	100	101
Representation			11	12	42	21	41	61	62
Source Code			0010	0011	1001	0100	1000	1100	1101

How Come this is Compression?!

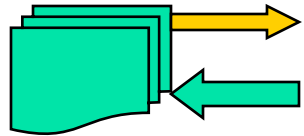
- The hope is:
 - If the bit sequence is long enough, eventually the fixed length code words will be shorter than the length of subsequences they represent.
- When applied to English text
 - Lempel-Ziv achieves approximately 55%
 - Huffman coding achieves approximately 43%

Encoding idea Lempel Ziv Welch-LZW

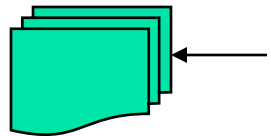
Assume we have just read
a **segment** w from the text.
 a is the next symbol.



If wa is not in the dictionary,

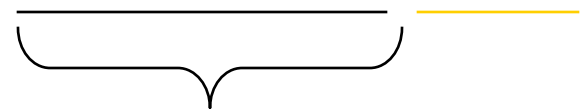


Write the index of w in the **output file**.
Add wa to the dictionary, and set $w \leftarrow a$.



wa is in the dictionary,

Process the next symbol with segment wa .



LZ Encoding example

Dictionary

Initial

a 0
b 1
c 2

Consider the input message:

Input string: a a b a a c a b c a b

LZ encoding process:

Input: a a b a a c a b c a b

a a aa not in dictionary, output 0 add aa to dictionary

output

update

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

aabaacabcb

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

LZ Decoding example

Dictionary

Initial

a	0
b	1
c	2

Input update

0

0

1

3

2

4

7

1

aa 3

ab 4

ba 5

aac 6

ca 7

abc 8

Output String:

a ? Output a

a a ! output a determines ? = a, update aa

a a b . output 1 determines !=b, update ab

a a b a a .

a a b a a c .

a a b a a c a b .

a a b a a c a b c a .

a a b a a c a b c a b

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