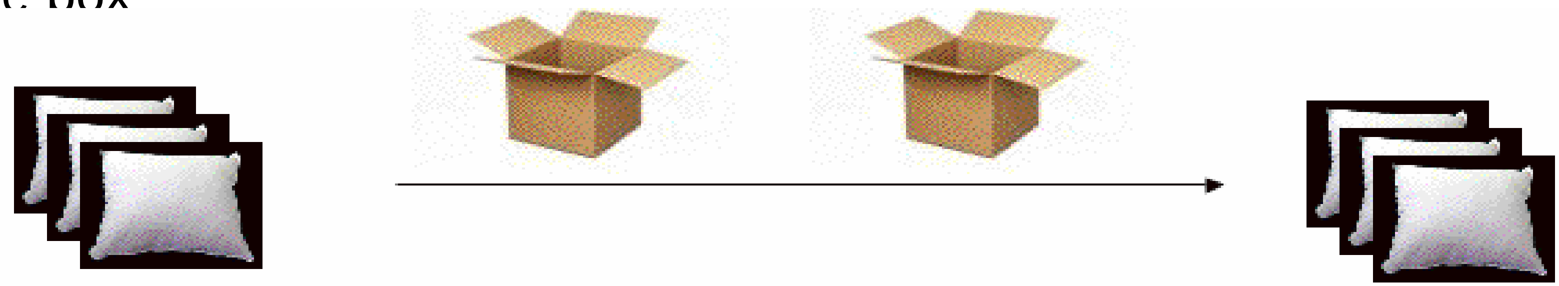
Entropy, Coding and Data Compression

Data vs. Information

- “yes,” “not,” “yes,” “yes,” “not” “not” ...
- In ASCII, each item is $3 \cdot 8 = 24$ bits ***of data***
- But if the only possible answers are “yes” and “not,” there is only one bit ***of information*** per item

Compression = Squeezing out the "Air"

- Suppose you want to ship pillows in boxes and are charged by the size of the box



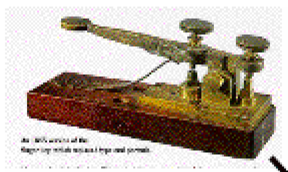
- To use as few boxes as possible, squeeze out all the air, pack into boxes, fluff them up at the other end
- Lossless data compression = pillows are perfectly restored
- Lossy data compression = some damage to the pillows is OK (MP3 is a lossy compression standard for music)
- Loss may be OK if it is below human perceptual threshold
- Entropy is a measure of limit of **lossless** compression

Example: Telegraphy

Source English letters -> Morse Code

Sender: from Hokkaido

D → -..



-..

-.. → D

Receiver: in Tokyo

Coding Messages with Fixed Length Codes

- Example: 4 symbols, A, B, C, D
- A=00, B=01, C=10, D=11
- In general, with *n* symbols, codes need to be of length lg *n*, rounded up
- For English text, 26 letters + space = 27 symbols, length = 5 since $2^4 < 27 < 2^5$

(replace all punctuation marks by space)

Modeling the Message Source

Source $\longrightarrow$ Destination

- Characteristics of the stream of messages coming from the source affect the choice of the coding method
- We need a model for a source of English text that can be described and analyzed mathematically

Uniquely decodable codes

- **If any encoded string has only one possible source string producing it then we have unique decodablity**

- **Example of uniquely decodable code is the *prefix code***

Prefix Coding

- A **prefix code** is defined as a code in which ***no*** codeword is the prefix of some other code word.
- A prefix code is ***uniquely decodable.***

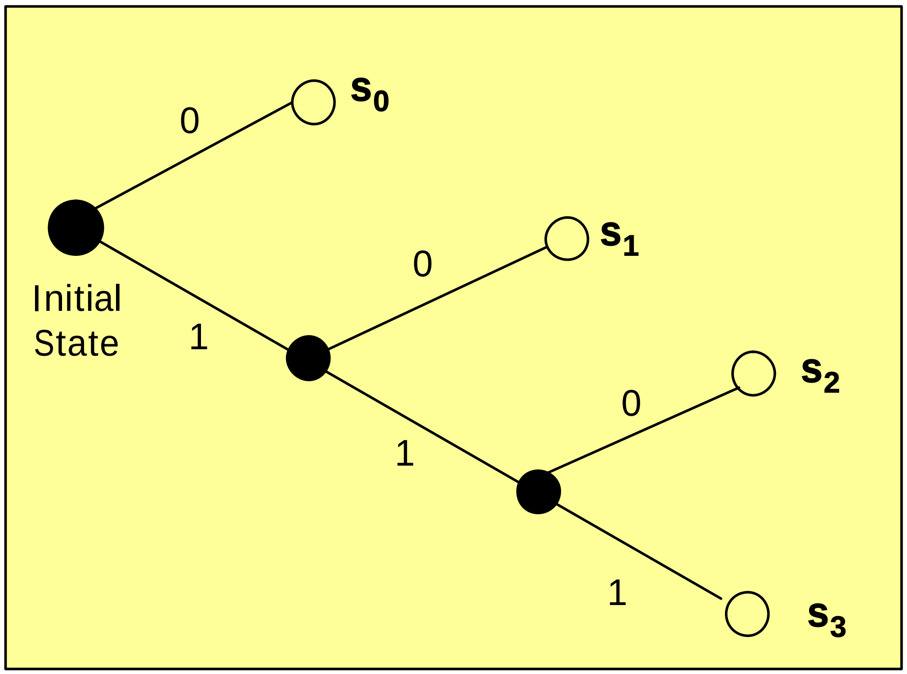
Source Symbol	Code A	Code B	Code C
	Symbol Codeword	Symbol Codeword	Symbol Codeword
s_0	0	0	0
s_1	1	10	01
s_2	00	110	011
s_3	11	111	0111

Prefix Code: Code B

Uniquely Decodable Codes: Code B, Code C

Decoding of a Prefix Code

Decision Tree for Code B



Code B	
Source Symbol s_k	Symbol Codeword c_k
s_0	0
s_1	10
s_2	110
s_3	111

- Example : Decode 1011111000
- Answer : $s_1 s_3 s_2 s_0 s_0$

Prefix Codes

Only one way to decode left to right when message received

Example 1

Symbol	A	B	C	D
Probability	.7	.1	.1	.1
Code	0	100	101	110

Received message:

$$\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{100}_{B}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{110}_{D}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{0}_{A}\underbrace{100}_{B}$$

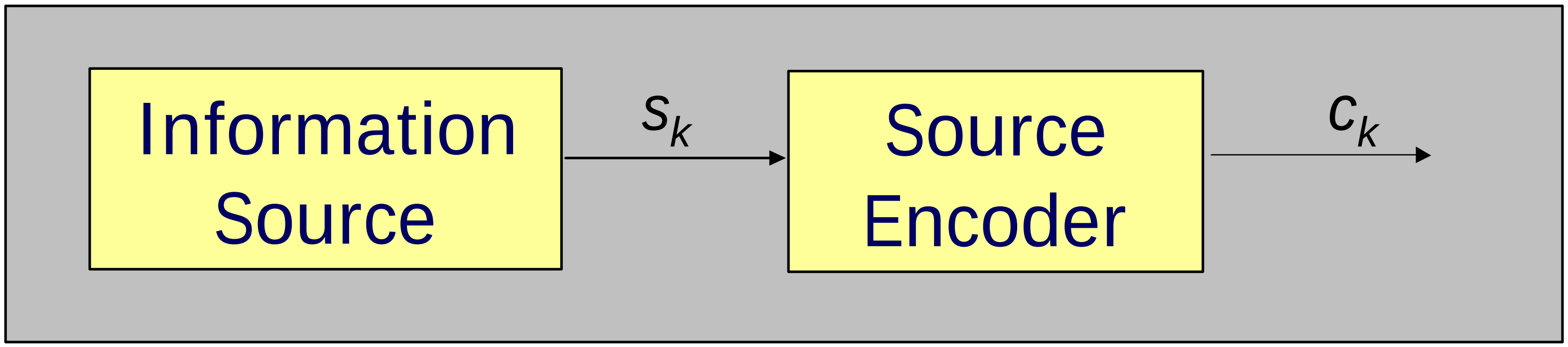
Prefix Codes

Example 2

Source Symbol s_k	Code E Symbol Codeword c_k
A	0
B	100
C	110
D	11

- IS CODE E A PREFIX CODE?
  - NO
  - WHY?
    - Code of D is a prefix to code of C

Average Code Length



- Source has K symbols
- Each symbol s_k has probability p_k
- Each symbol s_k is represented by a codeword c_k of length l_k bits
- **Average codeword length**

$$L = \sum_{k=0}^{K-1} p_k l_k$$

Shannon's First Theorem: The Source Coding Theorem

$$\boxed{L \geq \mathrm{H}(S)}$$

- The outputs of an information source cannot be represented by a source code whose average length is less than the source entropy

Average Code Length

Example

Average bits per symbol:

$L = .7 \cdot 1 + .1 \cdot 3 + .1 \cdot 3 + .1 \cdot 3 = 1.6$

bits/symbol (down from 2)

A	B	C	D
.7	.1	.1	.1
0	100	101	110

Another prefix code that is better

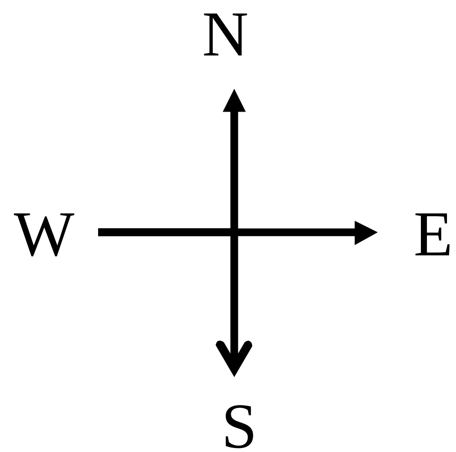
$L = .7 \cdot 1 + .1 \cdot 2 + .1 \cdot 3 + .1 \cdot 3 = 1.5$

A	B	C	D
.7	.1	.1	.1
0	10	110	111

Source Entropy Examples

Robot Example

- 4-way random walk



$$prob(x = S) = \frac{1}{2},\ prob(x = N) = \frac{1}{4}$$

$$prob(x = E) = prob(x = W) = \frac{1}{8}$$

$$H(X) = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{8} + \frac{1}{8}\log_2\frac{1}{8}) = 1.75bps$$

Source Entropy Examples

Robot Example

symbol k	p_k	*fixed-length codeword*	*variable-length codeword*
S	*0.5*	*00*	*0*
N	*0.25*	*01*	*10*
E	*0.125*	*10*	*110*
W	*0.125*	*11*	*111*

symbol stream : S S N W S E N N N W S S S N E S S

fixed length: **00 00 01 11 00 10 01 01 11 00 00 00 01 10 00 00** — 32bits

variable length: **0 0 10 111 0 110 10 10 111 0 0 0 10 110 0 0** — 28bits

4 bits savings achieved by VLC (redundancy eliminated)

Entropy, Compressibility, Redundancy

- Lower entropy <=> More redundant <=> More compressible
- Higher entropy <=> Less redundant <=> Less compressible
- A source of “yes”s and “not”s takes 24 bits per symbol but contains at most one bit per symbol of information

Entropy and Compression

- First-order entropy is theoretical minimum on code length when only frequencies are taken into account
- $L = .7 \cdot 1 + .1 \cdot 2 + .1 \cdot 3 + .1 \cdot 3 = 1.5$
- First-order Entropy = 1.353

A	B	C	D
.7	.1	.1	.1
0	10	110	111

- First-order Entropy of English is about 4 bits/character based on "typical" English texts

Bits

You are watching a set of independent random samples of X

You see that X has four possible values

P(X=A) = 1/4	P(X=B) = 1/4	P(X=C) = 1/4	P(X=D) = 1/4

So you might see output: BAACBADCDADDDA...

You transmit data over a binary serial link. You can encode each reading with two bits (e.g. A = 00, B = 01, C = 10, D = 11)

2 bits on average per symbol

0100001001001110110011111100...

Fewer Bits

Someone tells you that the probabilities are not equal

$P(X=A) = 1/2$	$P(X=B) = 1/4$	$P(X=C) = 1/8$	$P(X=D) = 1/8$

Is it possible...

...to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

Fewer Bits

Someone tells you that the probabilities are not equal

$P(X=A) = 1/2$	$P(X=B) = 1/4$	$P(X=C) = 1/8$	$P(X=D) = 1/8$

It's possible...

...to invent a coding for your transmission that only uses 1.75 bits on average per symbol. How?

A	**0**
B	**10**
C	**110**
D	**111**

(This is just one of several ways)

Fewer Bits

Suppose there are three equally likely values...

P(X=A) = 1/3	P(X=B) = 1/3	P(X=C) = 1/3

Here's a naïve coding, costing 2 bits per symbol

A	**00**
B	**01**
C	**10**

Can you think of a coding that would need only 1.6 bits per symbol on average?

In theory, it can in fact be done with 1.58496 bits per symbol.

Kraft-McMillan Inequality

$$\sum_{k=0}^{K-1} 2^{-l_k} \le 1$$

Source Symbol s_k	Code D	
	Symbol Codeword c_k	Codeword Length l_k
s_0	0	1
s_1	10	2
s_2	110	3
s_3	11	2

- If codeword lengths of a code satisfy the Kraft McMillan's inequality, then a prefix code with these codeword lengths ***can be*** constructed.
- For code D
  - $2^{-1}+ 2^{-2}+ 2^{-3}+ 2^{-2}=9/8$
  - This means that Code D **IS NOT A PREFIX CODE**

Use of Kraft-McMillan Inequality

- We may use it if the number of symbols are large such that we cannot simply by inspection judge whether a given code is a prefix code or not
- **WHAT Kraft-McMillan Inequality Can Do:**
  - It can determine that a given code IS NOT A PREFIX CODE
  - It can identify that a prefix code could be constructed from a set of codeword lengths
- WHAT Kraft-McMillan Inequality **Cannot** Do:
  - It cannot guarantee that a given code is indeed a prefix code

Example

Source Symbo l s_k	Code E	
	Symbol Codewor d c_k	Codeword Length l_k
s_0	0	1
s_1	100	3
s_2	110	3
s_3	11	2

- For code E
  - $2^{-1}+ 2^{-2}+ 2^{-3}+ 2^{-3=}1$
- IS CODE E A PREFIX CODE?
  - NO
  - WHY?
    - s_3 is a prefix to s_2

Code Efficiency ?

$$\boxed{h = \frac{\mathrm{H}(S)}{L}}$$

- An efficient code means ?→1

Examples

Source Symbol s_k	Symbol Probability p_k	Code I		Code II	
		Symbol Codeword c_k	Codeword Length l_k	Symbol Codeword c_k	Codeword Length l_k
s_0	1/2	00	2	0	1
s_1	1/4	01	2	10	2
s_2	1/8	10	2	110	3
s_3	1/8	11	2	111	3

- Source Entropy
  - $H(S) = 1/2\log_2(2) + 1/4\log_2(4) + 1/8\log_2(8) + 1/\log_2(8)$
    $= 1\ 3/4$ bits/symbol

Code I

$$L = 2 \times \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} \right) = 2$$

$$h = \frac{7/4}{2} = 0.875$$

Code II

$$L = \left(1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 3 \times \frac{1}{8} \right) = \frac{7}{4}$$

$$h = \frac{7/4}{7/4} = 1$$

For a Prefix Code

- Shannon's First Theorem $\boxed{\mathrm{H}(S) \le L < \mathrm{H}(S) + 1}$

$$L = \mathrm{H}(S) \qquad \textbf{if} \qquad p_k = 2^{-l_k} \; \forall k$$

What is the Efficiency ? ?

? =1

if $p_k \neq 2^{-l_k}$ for some $k \Rightarrow$?<1

However, we may increase efficiency by extending the source

Increasing Efficiency by Source Extension

- By extending the source we may potentially increase efficiency
- The drawback is
  - Increased decoding complexity

$$H(S^n) \le \bar{L}_n < H(S^n) + 1$$

$$nH(S) \le L_n < nH(S) + 1$$

$$H(S) \le \frac{L_n}{n} < H(S) + \frac{1}{n}$$

$$\eta = \frac{H(S)}{L_n / n}$$

$\eta \to 1$ when

$n \to \infty$

Extension of a Discrete Memoryless Source

- Treats Blocks of *n* successive symbols

Information Source

$$S = \{ s_0, s_1, \ldots, s_{K-1} \}$$

$$\Pr\{s_k\} = p_k, k = 0, 1, \ldots, K-1$$

$$\sum_{k=0}^{K-1} p_k = 1$$

Extended Information Source

$$S^n = \{ s_0, s_1, \ldots, s_{K^n-1} \}$$

$$\Pr\{s_i\} = q_i, i = 0, 1, \ldots, K^n - 1$$

$$\sum_{i=0}^{K^n-1} p_i = 1$$

Example 2

- $S=\{s_0,s_1,s_2\}$, $p_0=1/4$, $p_1=1/4$, $p_2=1/2$
- $H(S)=(1/4)\log_2(4)+ (1/4)\log_2(4)+ (1/2)\log_2(2)$

  $H(S)=3/2$ bits

Second-Order Extended Source

Symbols of S^2	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
Sequence of Symbols from S	$s_0\,s_0$	$s_0\,s_1$	$s_0\,s_2$	$s_1\,s_0$	$s_1\,s_1$	$s_1\,s_2$	$s_2\,s_0$	$s_2\,s_1$	$s_2\,s_0$
$P\{s_i\}$, $i=0,1,\ldots,8$	1/16	1/16	1/8	1/16	1/16	1/8	1/8	1/8	1/4

By Computing: $H(S^2)=3$ bits

Example 3

- Calculate the English of English language if
    1. All alphabet letters are equally probable
    2. For a, e, o, t $P\{s_k\}=0.1$

       For h, i, n, r, s $P\{s_k\}=0.07$

       For c, d, f, l, m, p, u, y $P\{s_k\}=0.02$

       For b, g, j, k, q, v, w, x, z $P\{s_k\}=0.01$

1. $H(S)=4.7$ bits
2. $H(S)=4.17$ bits

- Source Encoding
  - ***Efficient*** representation of information sources
- Source Coding Requirements
  - Uniquely Decodable Codes
- Prefix Codes
  - No codeword is a prefix to some other code word

Code Efficiency

$$\boxed{\eta = \frac{\mathrm{H}(S)}{\bar{L}}}$$

Kraft's Inequality

$$\boxed{\sum_{k=0}^{K-1} 2^{-l_k} \leq 1}$$

Source Coding Theorem

$$\boxed{\mathrm{H}(S) \leq \bar{L} < \mathrm{H}(S) + 1}$$

Source Coding Techniques

1. Huffman Code.

2. Two-path Huffman Code.

3. Lemple-Ziv Code.

4. Shannon Code.

5. Fano Code.

6. Arithmetic Code.

Source Coding Techniques

1. Huffman Code.

2. Two-path Huffman Code.

3. Lemple-Ziv Code.

4. Shannon Code.

5. Fano 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

Huffman Coding: Example 1

- Compute the Huffman Code for the source shown

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

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

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	

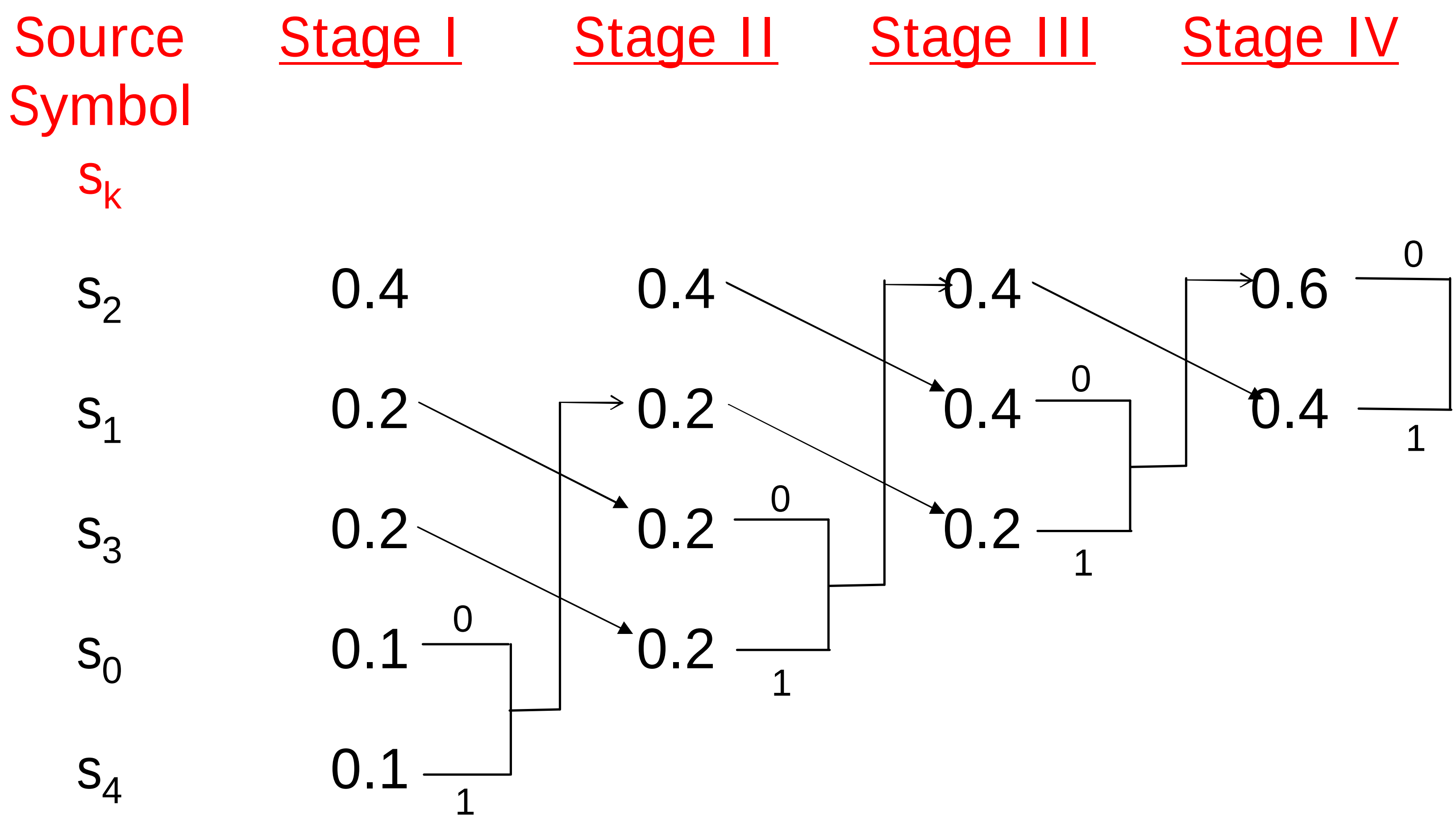
Solution A

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

Source Symbol s_k	Stage I	Stage II	Stage III	Stage IV
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 s_k
Stage I
Stage II
Stage III
Stage IV
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
0
1
0
1
0
1
0
1

Solution A

Source Symbol s_k	Stage I	Stage II	Stage III	Stage IV	Code
s_2	0.4	0.4	0.4	0.6	**00**
s_1	0.2	0.2	0.4	0.4	**10**
s_3	0.2	0.2	0.2		**11**
s_0	0.1	0.2			**010**
s_4	0.1				**011**

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

$$\boxed{\mathrm{H}(S) = 2.12193}$$

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

$$\boxed{\mathrm{H}(S) \le \bar{L} < \mathrm{H}(S) + 1}$$

THIS IS NOT THE ONLY SOLUTION!

Alternate Solution B

Source Symbol s_k	Stage I	Stage II	Stage III	Stage IV	Code
s_2	0.4	0.4	0.4	0.6	**1**
s_1	0.2	0.2	0.4	0.4	**01**
s_3	0.2	0.2	0.2		**000**
s_0	0.1	0.2			**0010**
s_4	0.1				**0011**

Alternative 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

$$\boxed{\mathrm{H}(S) = 2.12193}$$

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

$$\boxed{\mathrm{H}(S) \le \bar{L} < \mathrm{H}(S) + 1}$$

What is the difference between the two solutions?

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

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

- Solution A
  - s^2=0.16
- Solution B
  - s^2=1.36

Source Coding Techniques

1. Huffman Code.

2. Two-path Huffman Code.

3. Lemple-Ziv Code.

4. Shannon Code.

5. Fano Code.

6. Arithmetic Code.

Source Coding Techniques

2. Two-path 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-path Huffman Code.

Example

Consider the input: ABABABABABACADABACADABACADABACAD

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	1.125	111	000

Source Coding Techniques

1. Huffman Code.

2. Two-path Huffman Code.

3. **Lemple-Ziv Code.**

4. Shannon Code.

5. Fano 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 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							
Representation									
Encoding									

Lempel-Ziv Coding Example

$\boxed{0\ 0}\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ldots$

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

Lempel-Ziv Coding Example

$\boxed{0\ 0}\ \boxed{0\ 1}$ 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									

Lempel-Ziv Coding Example

$\boxed{0\ 0\ 0\ 1}\ \boxed{0\ 1\ 1}\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ldots$

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

Lempel-Ziv Coding Example

$\boxed{0\ 0\ 0\ 1\ 0\ 1\ 1}\ \boxed{1\ 0}\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ldots$

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

Lempel-Ziv Coding Example

$\boxed{0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0}\ \boxed{0\ 1\ 0}\ 1\ 0\ 0\ 1\ 0\ 1\ldots$

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

$\boxed{0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0}\ \boxed{1\ 0\ 0}\ 1\ 0\ 1\ldots$

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

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			1**1**	1**2**	4**2**	2**1**	4**1**	6**1**	6**2**
Source Code			001**0**	001**1**	100**1**	010**0**	100**0**	110**0**	110**1**

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%