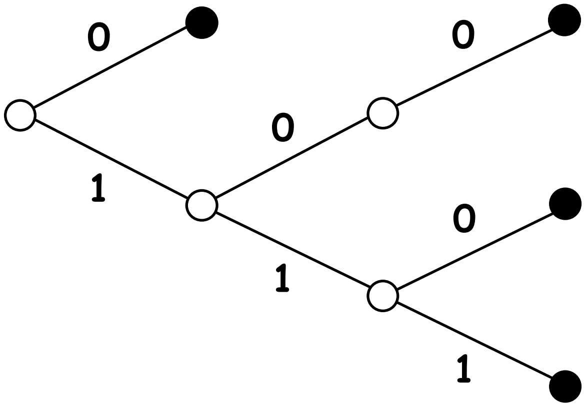
Information Coding and Compression



Hiroki Sayama
sayama@binghamton.edu

From measurement to coding

- **So far:**

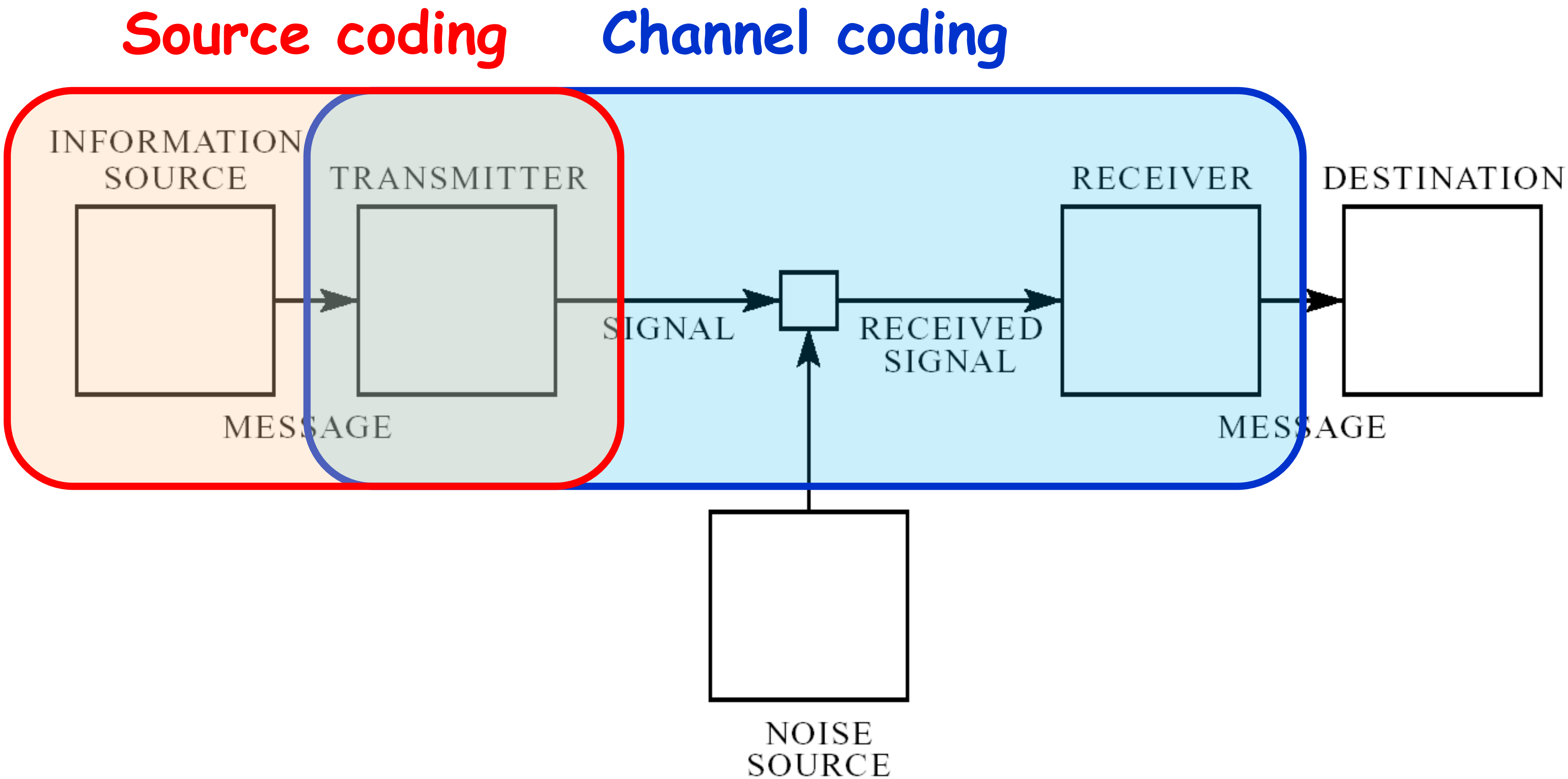
  We measured the amount of information produced by an event or a stochastic system

- **Today:**

  We will discuss how to represent the produced information effectively using symbols (information coding)

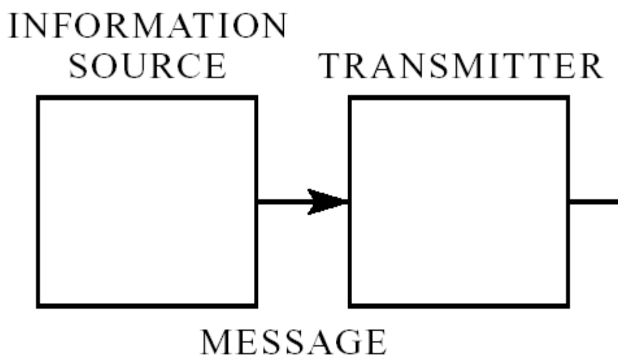
Information Coding

Shannon's model



From C. E. Shannon (1948) A mathematical theory of communication.
The Bell Sys. Tech. J. 27: 379-423, 623-656.

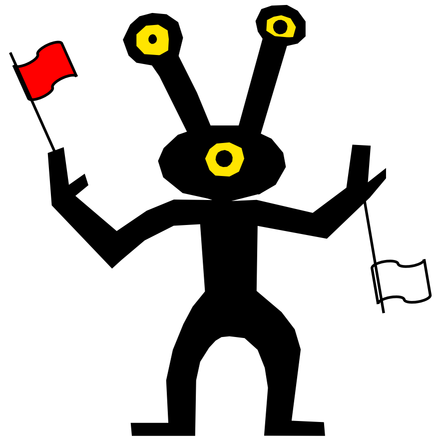
Source coding

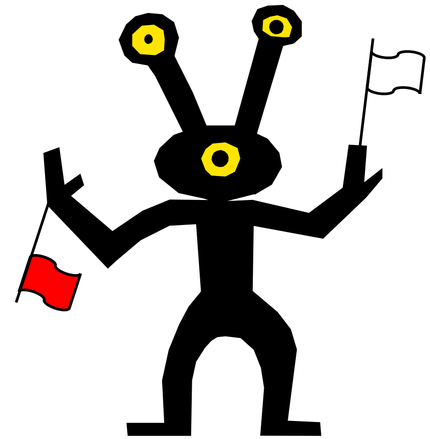


- Representing the behavior of a stochastic system (information source) using symbols
  - Temporal behavior (e.g., text, sound)
  - Spatial behavior (e.g., image)
  - Spatio-temporal behavior (e.g., video)

Example

Stochastic system

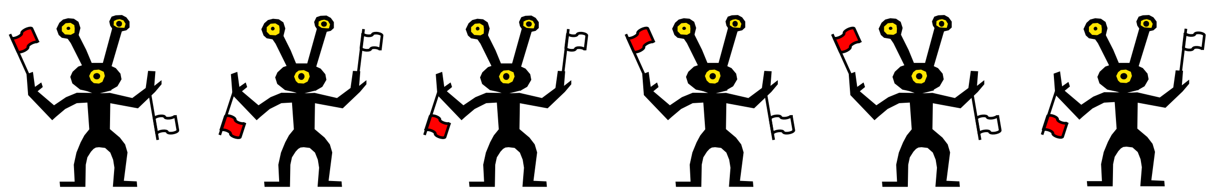




Symbols

1 0   0 1

System's behavior



Representation 1 0   0 1   0 1   1 0   1 0   0 1

Terminologies (1)

- **Source alphabet S:** A set of symbols that can arise in the information source
- **Target alphabet T:** A set of symbols that can be used in a code word
- **Code word:** A sequence of symbols in T used to represent a symbol in S

Terminologies (2)

- **Code:** A mapping of each symbol in S to a code word made of symbols in T
- **Encoding:** Conversion of a symbol in S to a code word made of symbols in T
- **Decoding:** Conversion of a code word made of symbols in T to a symbol in S

Terminologies (3)

- **Fixed-length code:**

  A code in which the length of code words are always the same

- **Variable-length code:**

  A code in which the length of code words vary depending on which symbol in S is encoded

Example of fixed-length codes

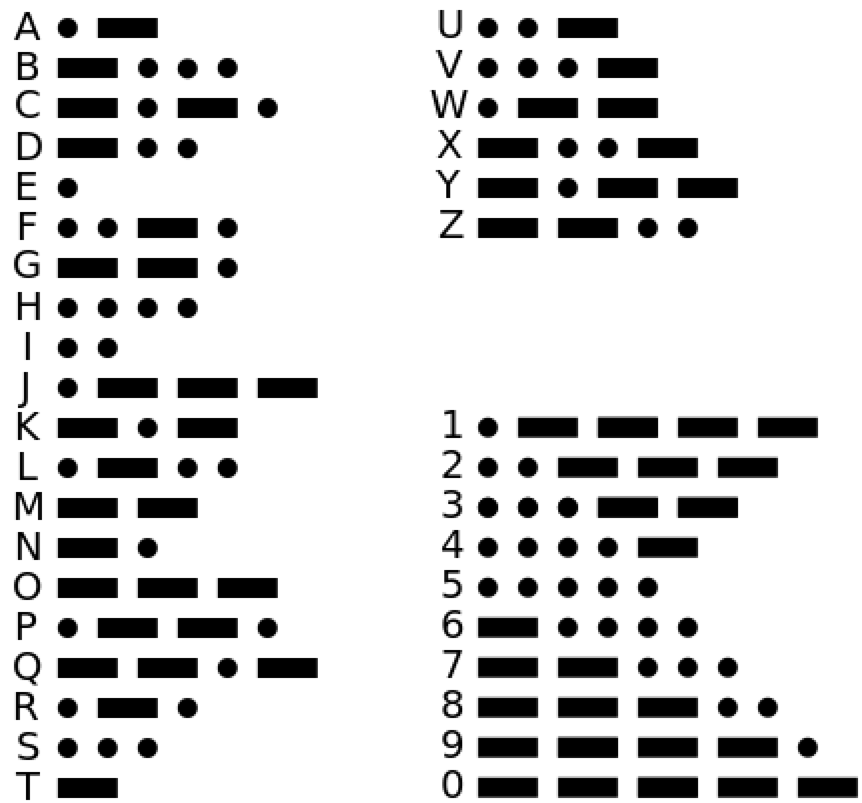
- **ASCII code**

USASCII code chart

Bits $b_7 b_6 b_5$ →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Example of variable-length codes

- International Morse code
- UTF-8
- Most of image, audio, and video formats



Very important for data compression

Exercise

- What is the minimal length of code words when n different symbols in S are to be encoded using a fixed-length code using r symbols?

- When is such a code most efficient?
  - In terms of the relationship between n and r

Exercise

(1) abdcdabccabdcdba

(2) acdacabcaabaabad

- Encode each of the above sequences using the following codes:

Fixed length: a→00, b→01, c→10, d→11

Variable length: a→0, b→10, c→110, d→111

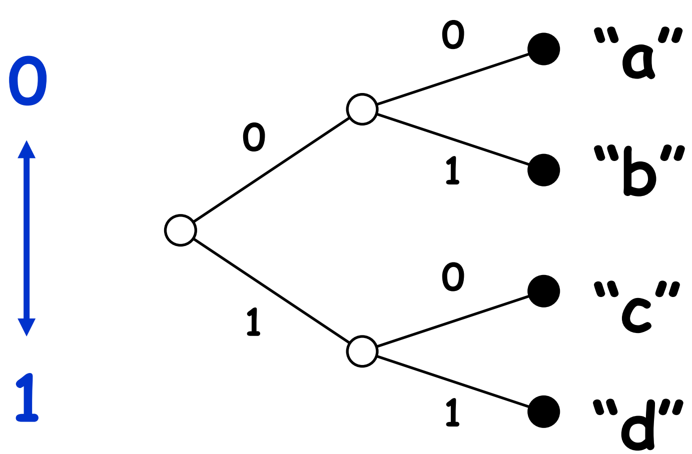
Code Trees, Prefix Codes and Kraft's Inequality

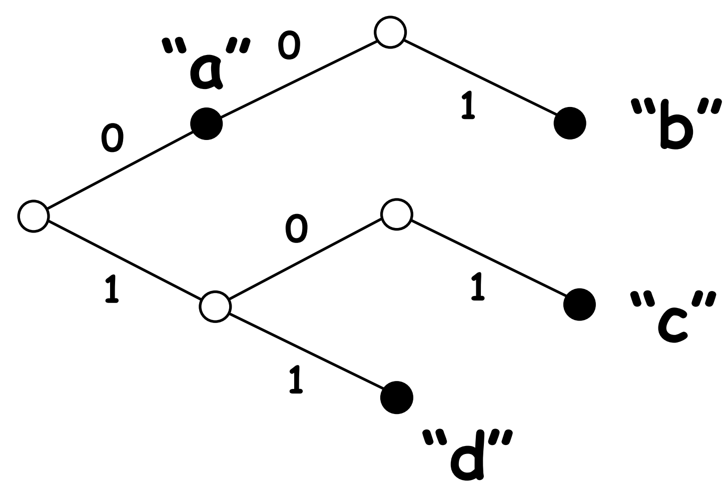
Code tree

- **A tree used to decode the provided code words**

a→00, b→01, c→10, d→11

a→0, b→001, c→101, d→11





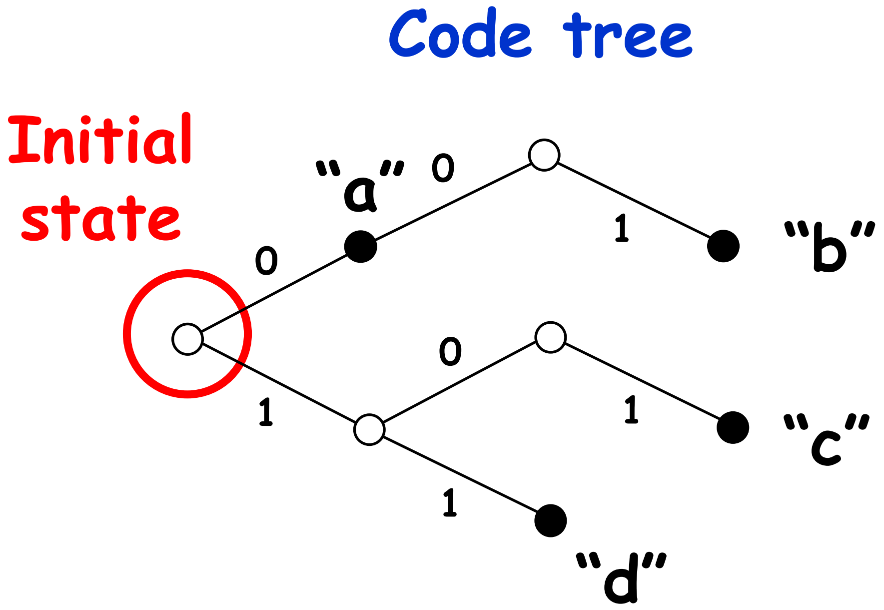
Black nodes: original symbols (= code words)

Exercise

- Decode the following input using the code tree provided below

Input

0011111



Exercise

- Create code trees for the following:
  - $a \to 0$, $b \to 001$, $c \to 101$, $d \to 11$
  - $a \to 1$, $b \to 100$, $c \to 101$, $d \to 11$
  - $a \to 01$, $b \to 0100$, $c \to 0111$, $d \to 010000$

Uniquely decodable code

- A code is called uniquely decodable if and only if any finite sequence of code words generated by it can be decoded to a unique sequence of original symbols with no ambiguity
  - Decoding may be ambiguous in the middle of the sequence, but not at the end

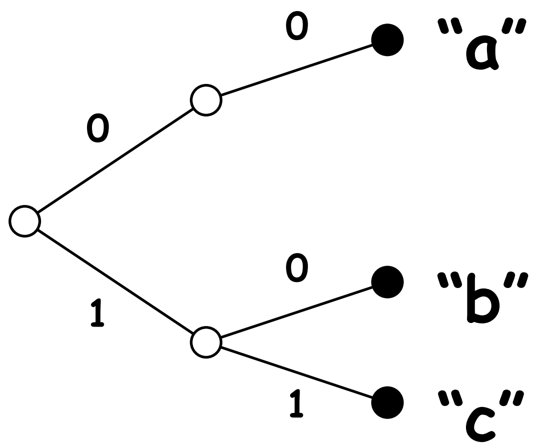
Codes must be uniquely decodable in order to be useful

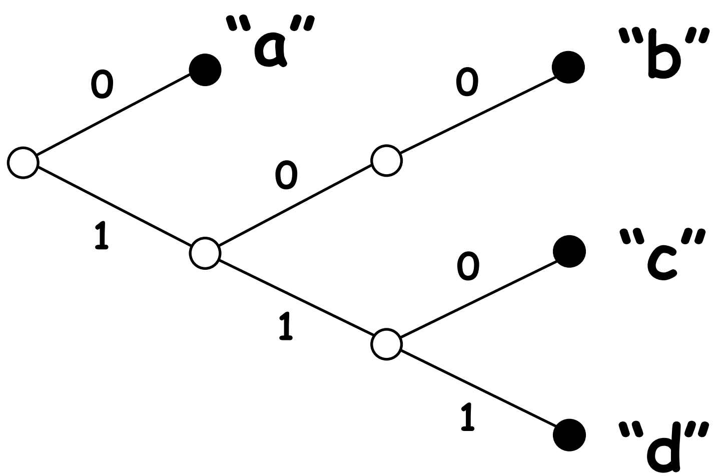
Exercise

- Are these codes uniquely decodable?
  - $a \rightarrow 0$, $b \rightarrow 001$, $c \rightarrow 101$, $d \rightarrow 11$
  - $a \rightarrow 1$, $b \rightarrow 100$, $c \rightarrow 101$, $d \rightarrow 11$
  - $a \rightarrow 01$, $b \rightarrow 0100$, $c \rightarrow 0111$, $d \rightarrow 010000$

Prefix code

- A code whose code words appear only as “leaves” (end points) in its code tree





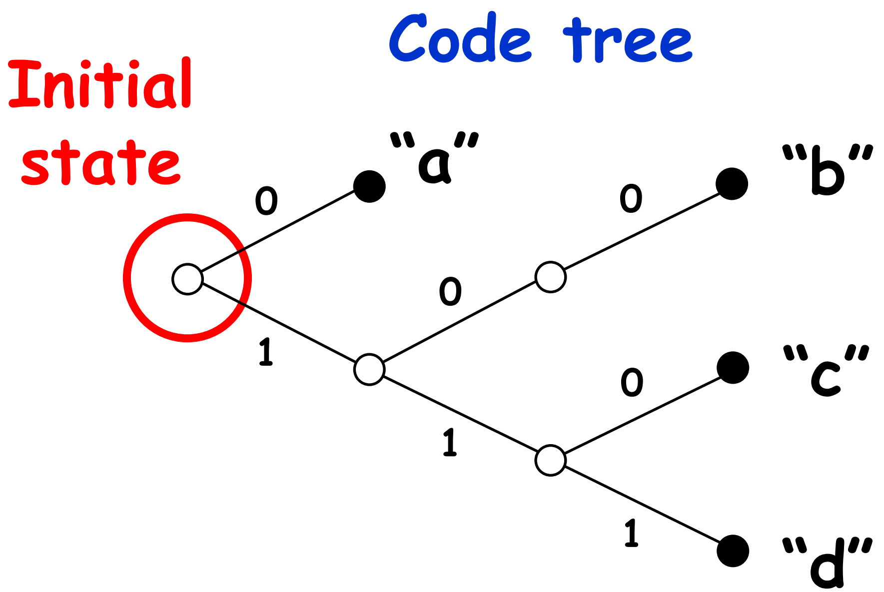
No code words in the middle of a branch
→ Uniquely, and <u>immediately</u> decodable

Exercise

- Decode the following input using the code tree provided below

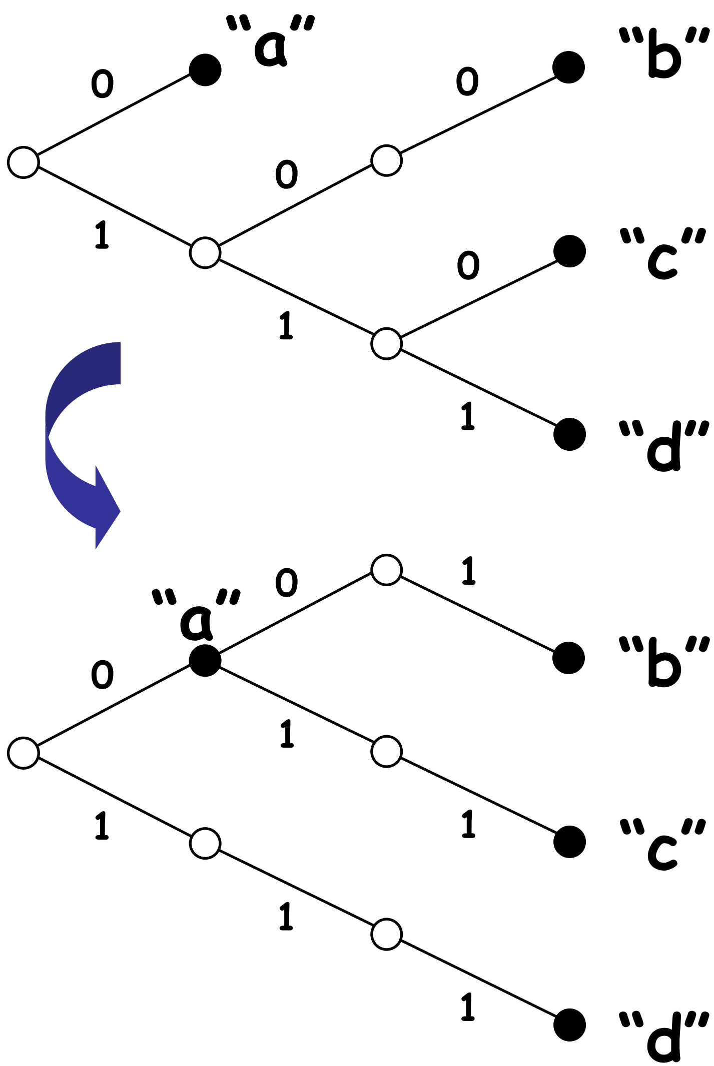
Input

1100100



Exercise

- A code generated by reverting the order of letters in each code word (e.g., 100→001) of a uniquely decodable code is known to be uniquely decodable too
- Check this with the example on the right
- Why is this the case?



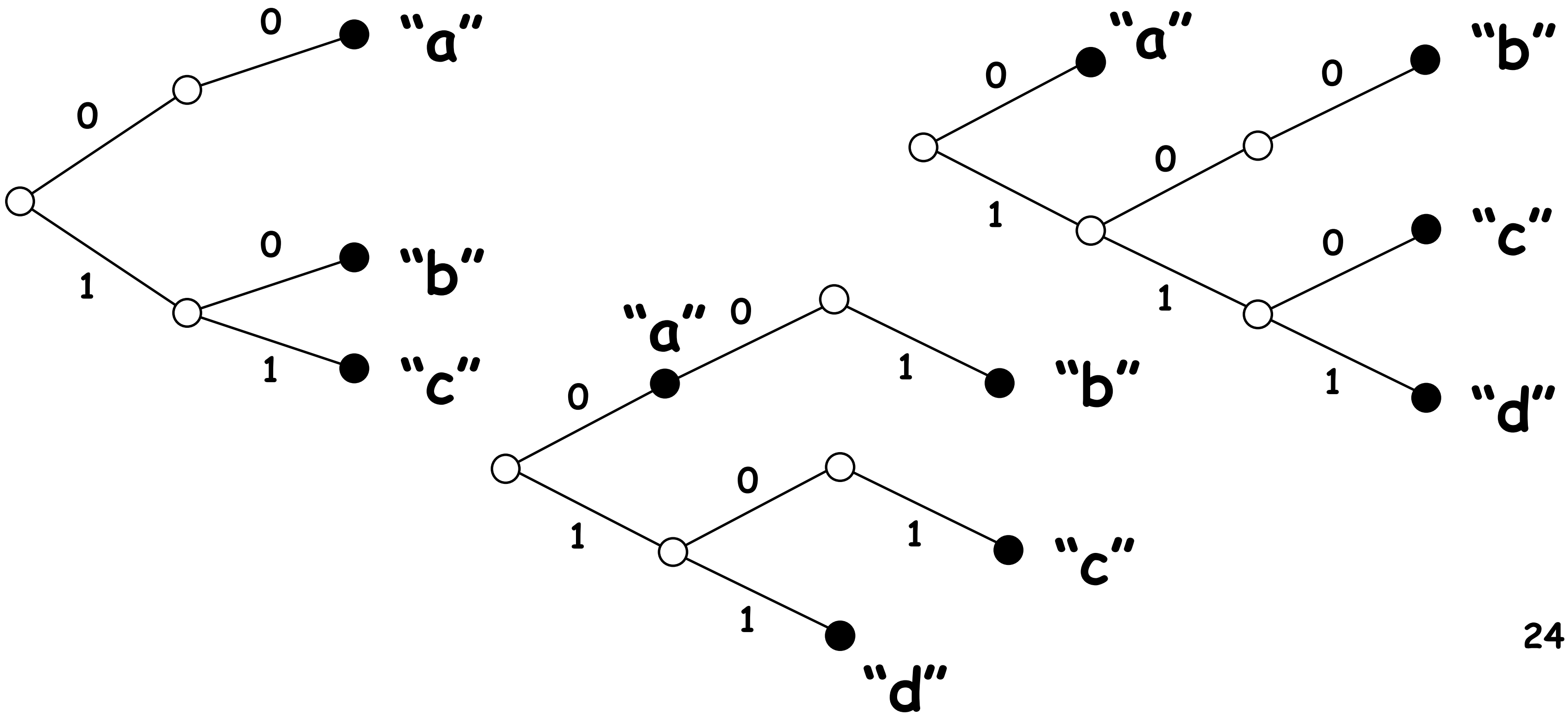
Kraft's inequality

- # of symbols in S: n
- # of symbols in T: r
- Length of code word: L_i $(i = 1 \sim n)$
- Necessary & sufficient condition for such a prefix code (or uniquely decodable code) to exist is:

$$\sum_i r^{-L_i} \leq 1$$

Exercise

- Check that Kraft's inequality holds for each of the following code trees



Exercise

- Determine whether it is possible to design a uniquely decodable binary code ($r = 2$) with each of the following code word lengths:

$\{L_i\}=\{1, 2, 3, 3\}$

$\{L_i\}=\{2, 2, 3, 3, 3, 4, 4\}$

$\{L_i\}=\{2, 2, 3, 3, 3, 3, 4\}$

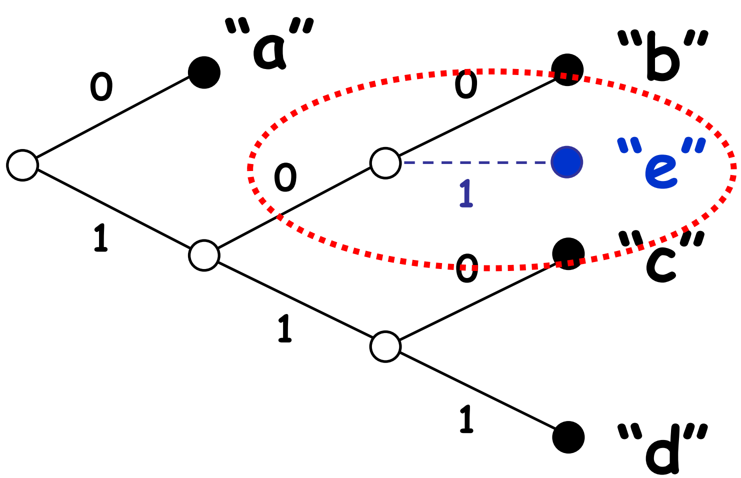
Complete code

Kraft's inequality holds with equality

⇕

The code is "complete"

(i.e., there are no more unused leaves)



Data Compression and Its Limit

Data compression by coding

How can one design an optimal code tree to achieve the shortest representation of the behavior of a stochastic system?

Average length and compression

$$L = \Sigma_i \, p_i \, L_i$$

(p_i: probability of i-th symbol in S)

- Data compression is to reduce the average code word length L by optimizing code word lengths { L_i } given the probability distribution of original symbols { p_i }

FYI: Lossless and lossy compression

- Here we focus on lossless compression that does not discard any information in the original data

- Lossy compression is also used very often in real-world applications
  - Exploits properties and limitations of human perception/cognition
  - E.g. audio/visual data compression

How can we compress the data?

$$L = \Sigma_i \; p_i \; L_i$$

(p_i: probability of i-th symbol in S)

- Basic idea:

  Assign smaller L_i to larger p_i

How much can we compress the data?

$$L_{min} = \min_{\{L_i\}\text{ that satisfies }\sum_i r^{-L_i} \le 1} \sum_i p_i L_i$$

Discrete optimization problem with integer $\{L_i\}$ → Very difficult to solve

Replacing $\{L_i\}$ with continuous variables $\{x_i\}$ allows $\sum_i r^{-x_i} = 1$ to find the theoretical lower bound of L_{min}

Exercise

- Find the lower bound of L_{min} using the Lagrange multiplier

Quantity to minimize:

$$\Sigma_i \, p_i \, x_i$$

Constraint: $\Sigma_i \, r^{-x_i} = 1$

Exercise (solution)

$$g(x_1, x_2, \dots x_n, x_{n+1})$$
$$= \Sigma_{i=1\sim n}\, p_i\, x_i + x_{n+1}\, (\Sigma_i\, r^{-x_i} - 1)$$

$$\partial g/\partial x_i = p_i - x_{n+1}\, r^{-x_i} \ln r = 0 \quad (i = 1\sim n)$$
$$\partial g/\partial x_{n+1} = \Sigma_i\, r^{-x_i} - 1 = 0$$

$$r^{-x_i} = p_i / (x_{n+1} \ln r) \quad (i = 1\sim n)$$

$$x_{n+1} = 1/\ln r$$

$$\underline{x_i = -\log_r p_i}$$

Lower bound of average code word length

$$L_{min} \geq \min \Sigma_i \, p_i \, x_i$$

$$= - \Sigma_i \, p_i \log_r p_i$$

Information entropy

Average code word length cannot be shorter than the entropy of the information source

In other words…

Information entropy is the amount of information the source is producing; you can't compress the data below that

(Data compression is only removing redundant part)

Huffman Coding

How to design an efficient code

- $\{p_i\}$ is given
- How to determine code word lengths $\{L_i\}$ to make the code most efficient?
- Continuous analog $x_i = -\log_r p_i$ serves as a good reference length
  - If $\{x_i\}$ are all integers, the average length can equal information entropy!

Exercise

- Design the most efficient code tree for each of the following probability distributions $\{p_i\}$ (with $r = 2$):

$\{p_i\} = \{1/4, 1/2, 1/4\}$

$\{p_i\} = \{1/4, 1/4, 1/8, 1/8, 1/8, 1/16, 1/16\}$

What if $\{x_i\}$ are not integers?

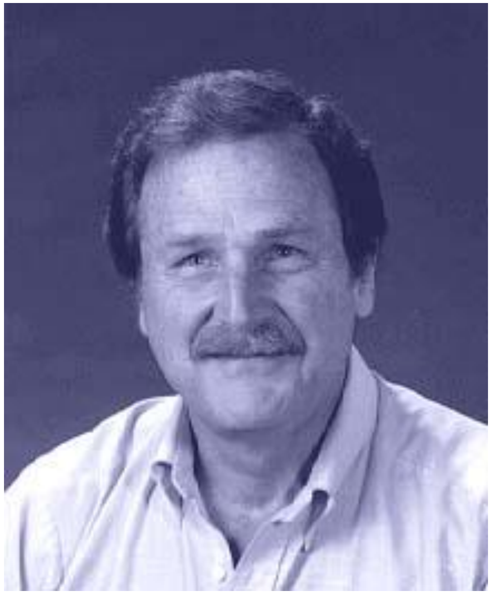
- It is no longer possible to make the average code word length equal the information entropy of the source
- However, there is an algorithm known for determining $\{L_i\}$ (in integers) of the most efficient code

Huffman coding

David A. Huffman (1925-1999)

"A method of the construction of minimum-redundancy codes"

Proc. Inst. Radio Eng. 40:1098-1101, 1952.



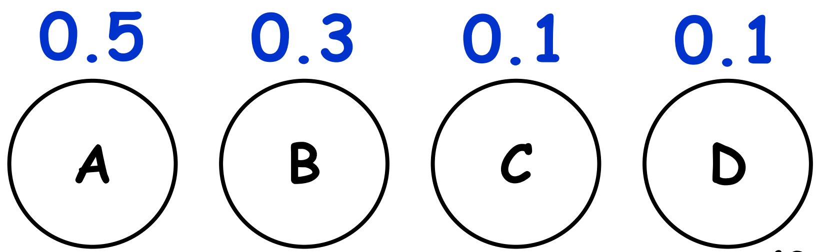
- Found this optimal algorithm when he was still a graduate student at MIT struggling with a final project of a course

Huffman coding

- An algorithm that generates the most efficient prefix code tree for a given probability distribution $\{p_i\}$ (with $r=2$)
- A prefix code generated by Huffman coding always achieves the <u>shortest average code word length</u>
  - It is also known that the achieved average length are always <u>less than the source's information entropy + 1</u>

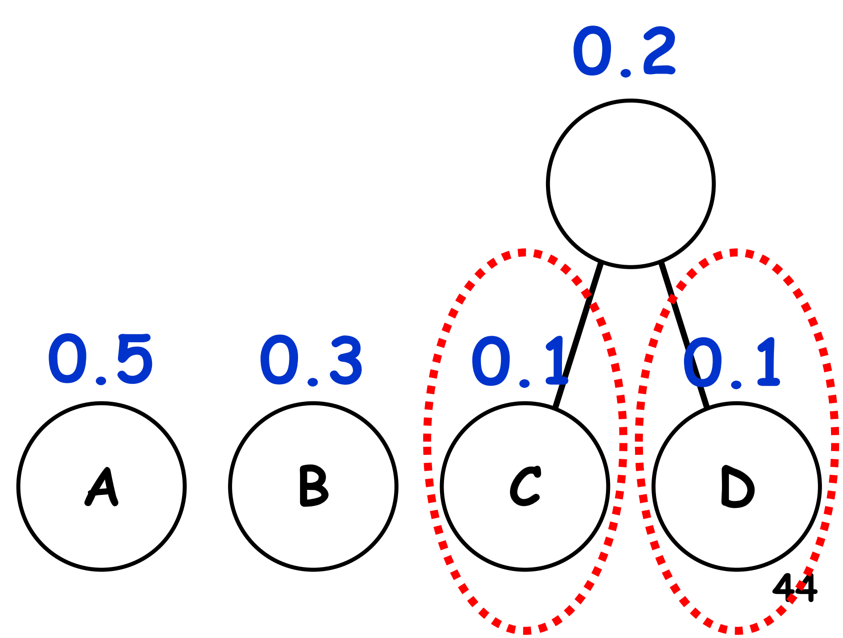
How Huffman coding works (1)

- Represent each letter in S as a node at the lowest level, with its probability



How Huffman coding works (2)

- Choose two nodes (without a parent node) that have the smallest probabilities
- Create a new parent node by adding the prob-abilities of two

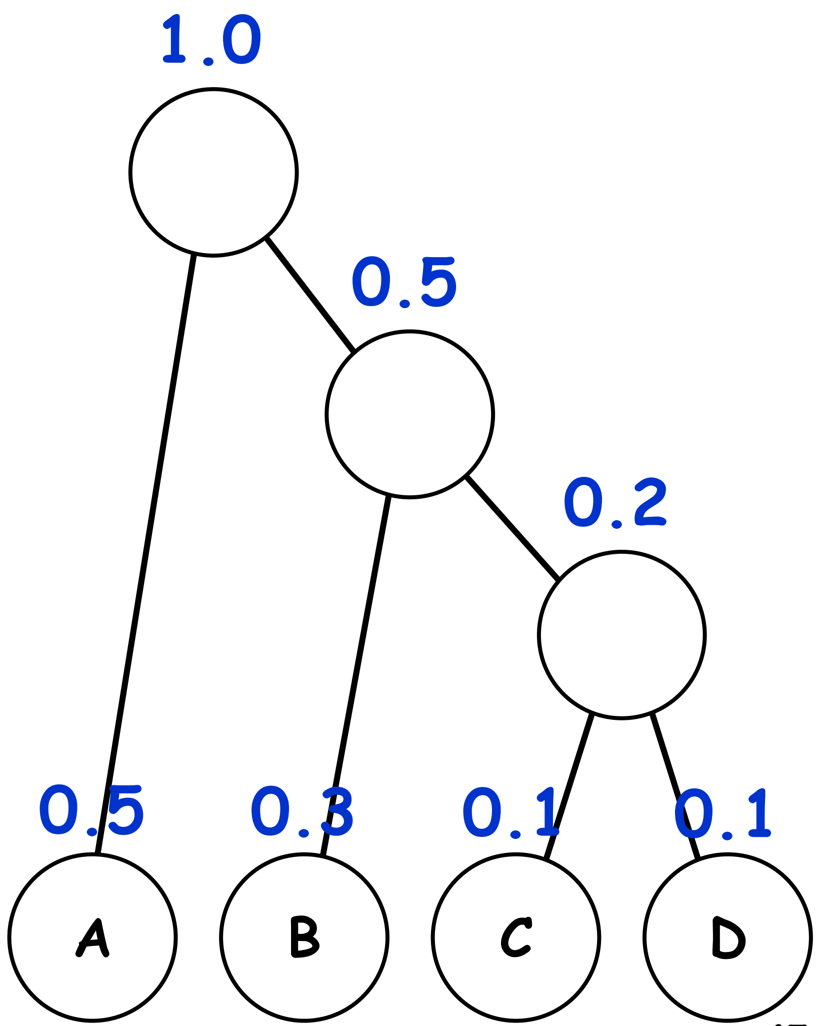


How Huffman coding works (3)

- Repeat it until there is only one tree
- Assign code words based on the generated tree

A: 0, B: 10, C: 110, D: 111

L_{min} = 0.5 * 1 + 0.3 * 2 + 0.1 * 3 + 0.1 * 3 = 1.7



Exercise

- Design the Huffman code tree for each of the following probability distributions $\{p_i\}$ (with $r = 2$):
- Compare their average code word lengths with theoretical lower bounds

$\{p_i\} = \{0.3, 0.25, 0.2, 0.1, 0.1, 0.05\}$

$\{p_i\} = \{0.4, 0.3, 0.1, 0.05, 0.05, 0.05, 0.05\}$

Exercise

- Compress the following text using Huffman coding (ignore space or line break):

CDEGF EBAEA
CEFAD AEFEA
BBEAC GDABC
AECAG CEAFG

Exercise

- Implement a program that generates a Huffman code for a given data set
- Apply the program to the text in the previous exercise to automatically generate an encoded text

Shannon's Source Coding Theorem

Average code word length and information entropy

- Information entropy of the source sets the lower bound of average code word length
- Can you reduce their difference?

Yes you can.

Shannon's source coding theorem

What it basically says

If you define an "extended information source" of S by combining its k consecutive symbols as a new symbol, then the average code word length (per original symbol) becomes arbitrarily close to its entropy as $k \rightarrow \infty$

Extended information source

- $S = \{s_i\}$ (prob. distribution: $\{p_i\}$)

- K-th order extended information source of S:

$$S^k = \{ s_{i_1} s_{i_2} \ldots s_{i_k} \}$$

$$\text{Prob. distribution: } \{ p_{i_1 i_2 \ldots i_k} \}$$

Example

- Source: Coin toss

  $S = \{ H, T \}$

- $S^2 = \{ HH, HT,$
  $TH, TT \}$

- $S^3 = \{ HHH, HHT,$
  $HTH, HTT,$
  $THH, THT,$
  $TTH, TTT \}$

Properties of S^k

- $| S^k | = | S |^k$

- $\Sigma \, p_{i_1 i_2 \ldots i_k} = 1$

- If S's behavior is independent and identically distributed (i.i.d.):

$$p_{i_1 i_2 \ldots i_k} = p_{i_1} \, p_{i_2} \cdots p_{i_k}$$

$$H(S^k) = k \, H(S)$$

Average code word length of S^k

- Huffman coding can be applied to S^k

→ How does its average code word length $L_{min}(k)$ behave as $k \to \infty$?

→ How does its average code word length per original symbol $L_{min}(k) / k$ behave as $k \to \infty$?

Exercise

- $S = \{A, B\}$, $\{p_i\} = \{1/4, 3/4\}$ (i.i.d.)
- Obtain probability distributions and their entropies of S^2, S^3 and S^4
- Construct their Huffman codes
- Calculate $L_{min}(k)$
- Calculate $L_{min}(k) / k$

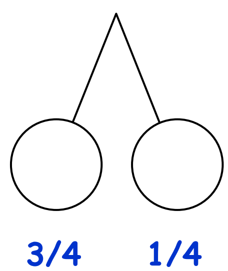
Results (k = 1, 2)

Entropy of original source

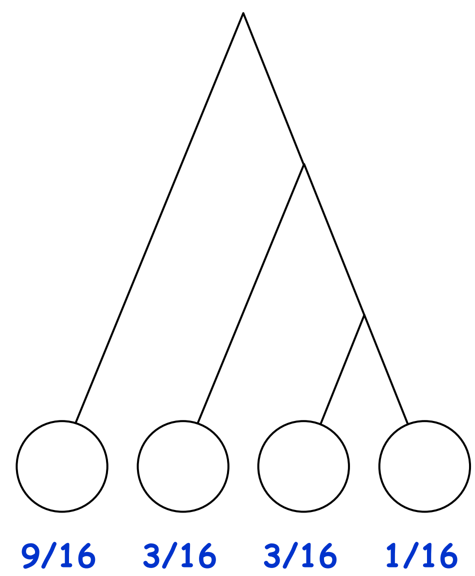
$H = 2 - 3/4 \log_2 3$

~ 0.811

k = 1



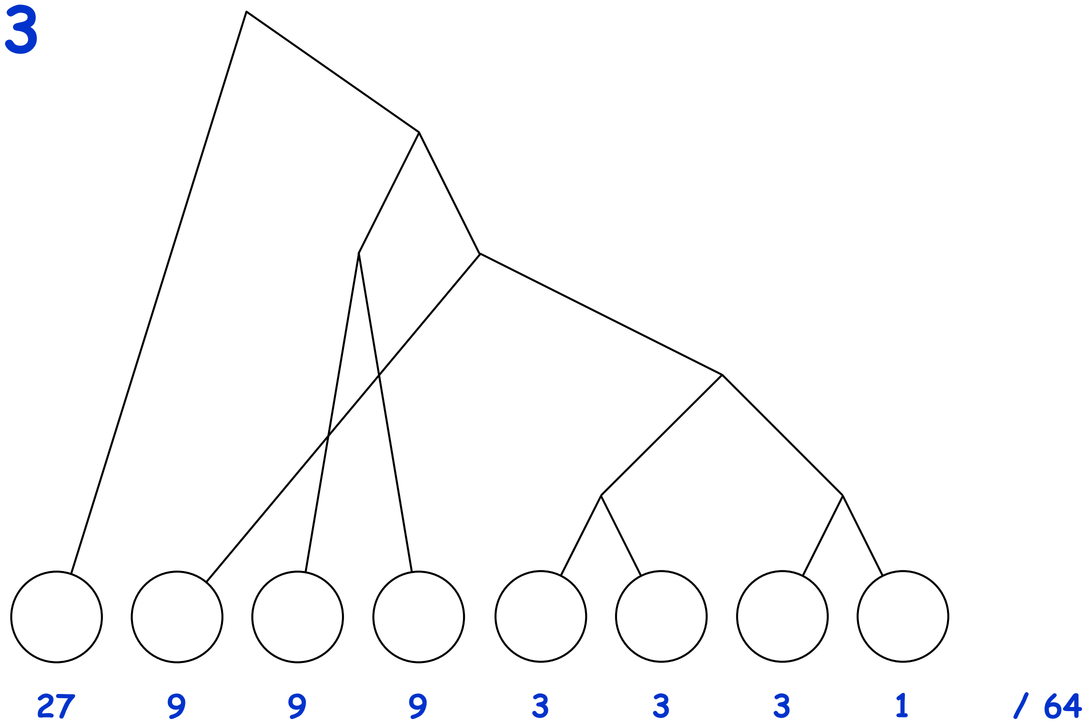
k = 2



	k = 1	k = 2
$L_{min}(k)$	1	1.688
$L_{min}(k)/k$	1	0.844

Results (k = 3)

k = 3

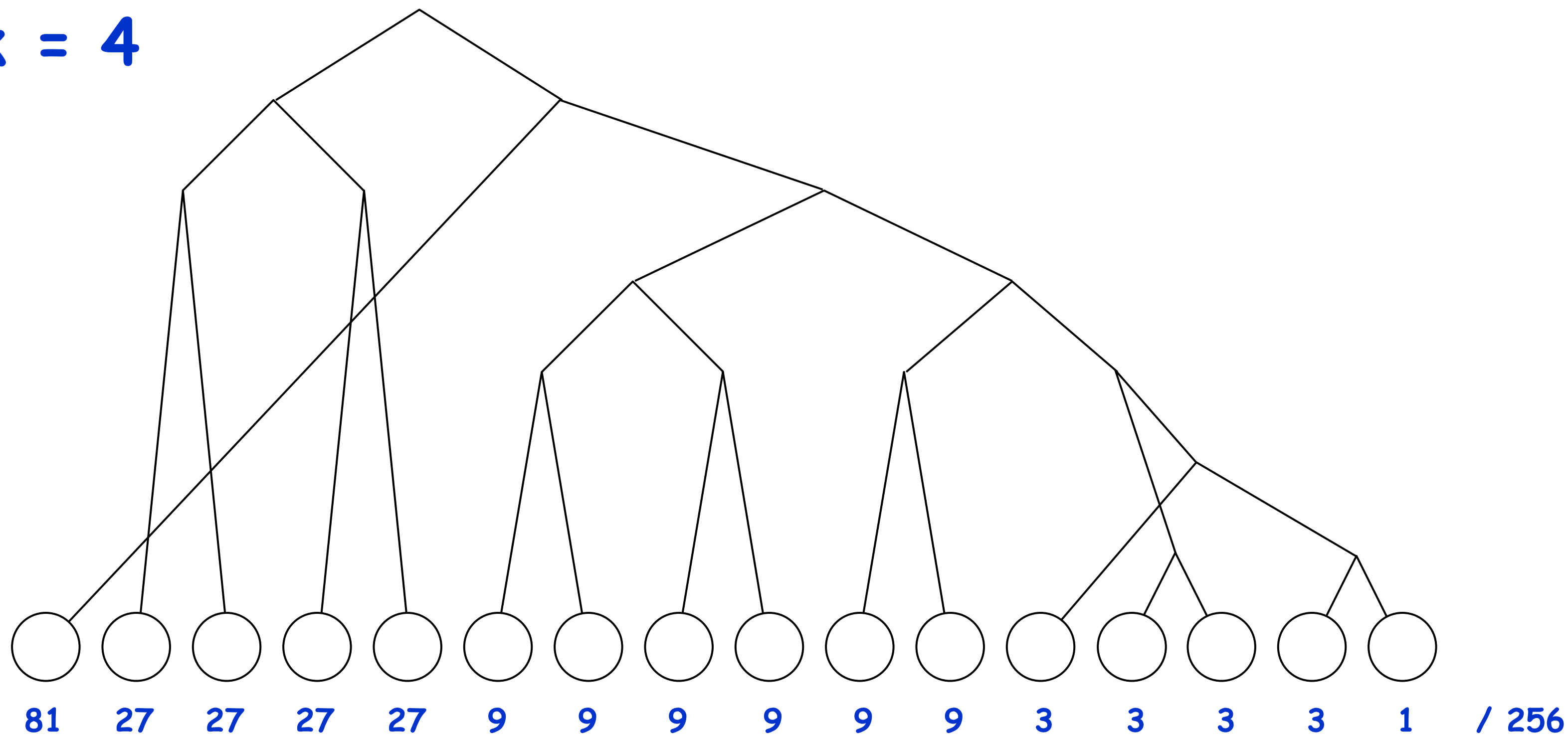


$L_{min}(k)$ 2.469

$L_{min}(k)/k$ 0.823

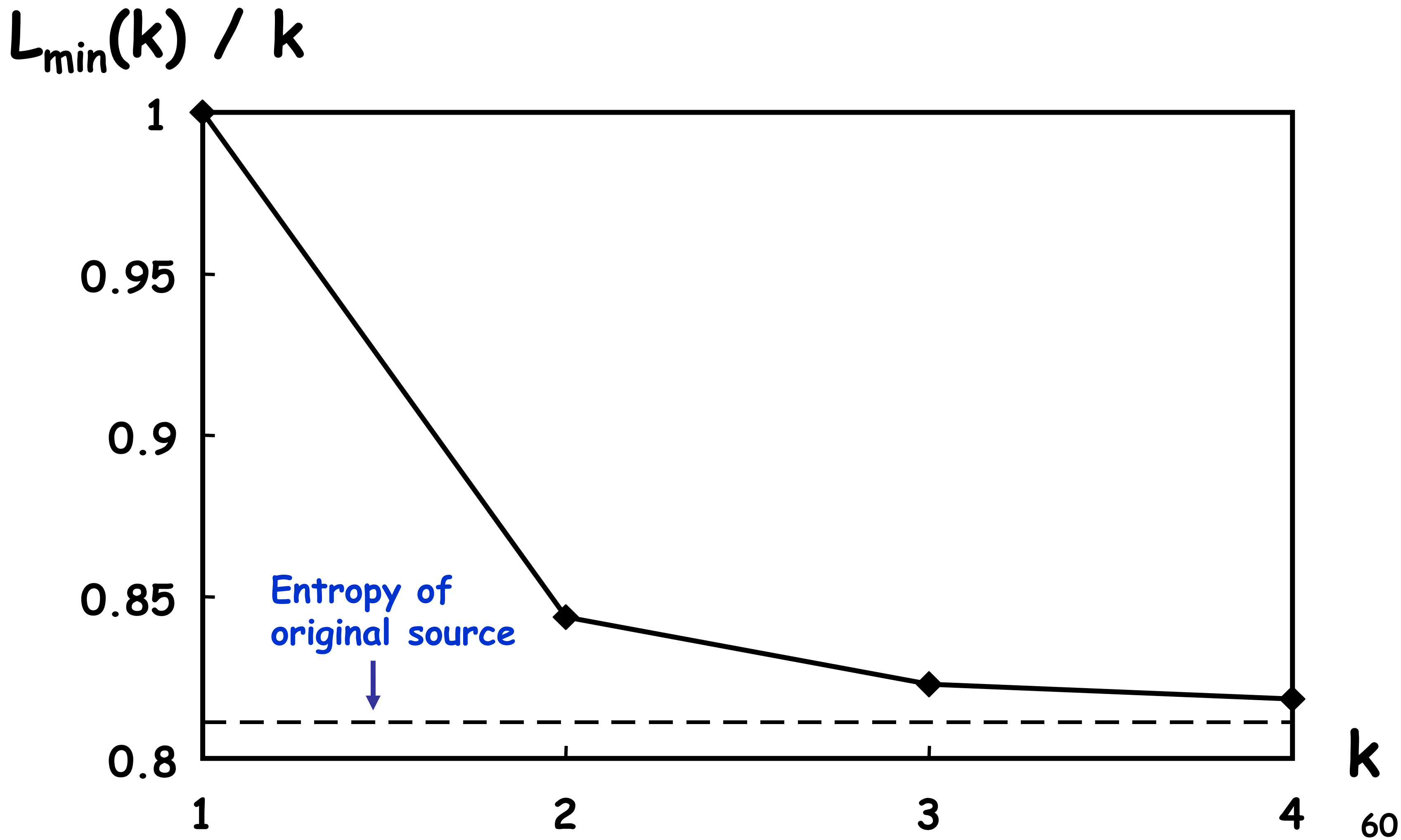
Results (k = 4)

k = 4



$L_{min}(k)$	3.273
$L_{min}(k)/k$	0.818

$L_{min}(k)$ / k approaching entropy



Why?

- Range of optimal average code word length:

$$H(X) \leq L_{min} < H(X) + 1$$

- k-th order extended version:

$$H(X^k) \leq L_{min}(k) < H(X^k) + 1$$

Why?

$$H(X^k) \leq L_{min}(k) < H(X^k) + 1$$

- Dividing both sides by k:

$$H(X^k)/k \leq L_{min}(k)/k < H(X^k)/k + \underline{1/k}$$

$L_{min}(k)/k$ converges to $\lim_{k\to\infty} H(X^k)/k$ as $k \to \infty$

Why?

$L_{min}(k)/k$ converges to $\lim_{k\to\infty} H(X^k)/k$ as $k \to \infty$

If X's behavior is i.i.d., then
$H(X^k) = k\,H(X)$
i.e., the above limit is always $H(X)$

Average code word length (per original symbol) becomes arbitrarily close to its entropy as $k \to \infty$!!

FYI: Generalization of entropy

- If the stochastic system's behavior is not i.i.d. (i.e., the values are correlated), then its information entropy is defined as follows:

$$\overline{H}(X) = \lim_{k \to \infty} H(X^k)/k$$

This is different from simple $H(X) = -\Sigma\, p_i \log p_i$

but the meaning is the same

Average # of bits needed to describe one event

Therefore…

If you define an "extended information source" of S by combining its k consecutive symbols as a new symbol, then the average code word length (per original symbol) becomes arbitrarily close to its entropy as $k \rightarrow \infty$

This holds for any information source

Exercise

- Apply the Huffman coding program to a k-th order extended information source of some real-world data and measure how compact the data is compressed
- Plot the length of compressed data for $k = 1 \sim 6$