Digital Communication Systems
ECS 452

Asst. Prof. Dr. Prapun Suksompong

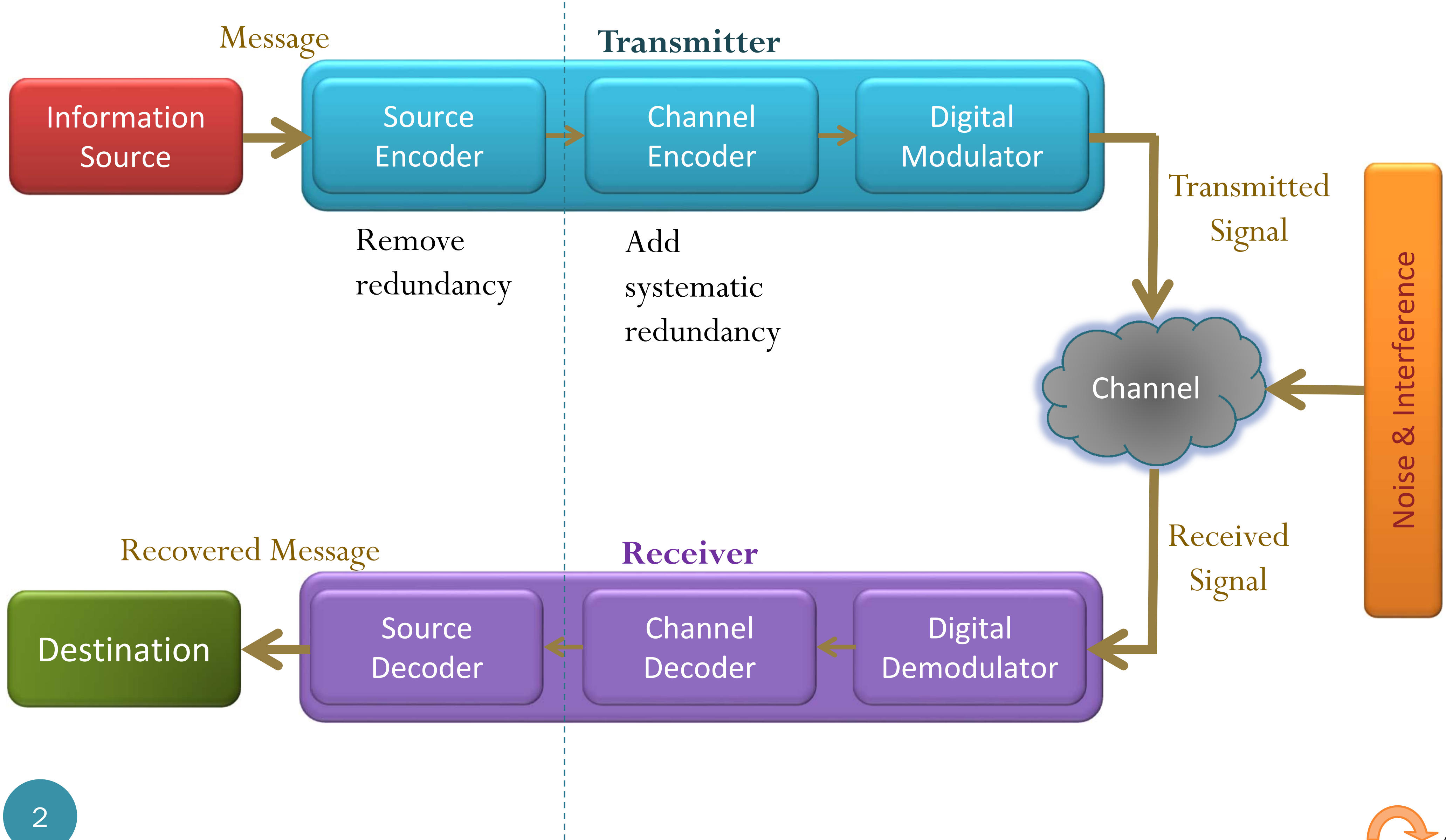
prapun@siit.tu.ac.th

2. Source Coding

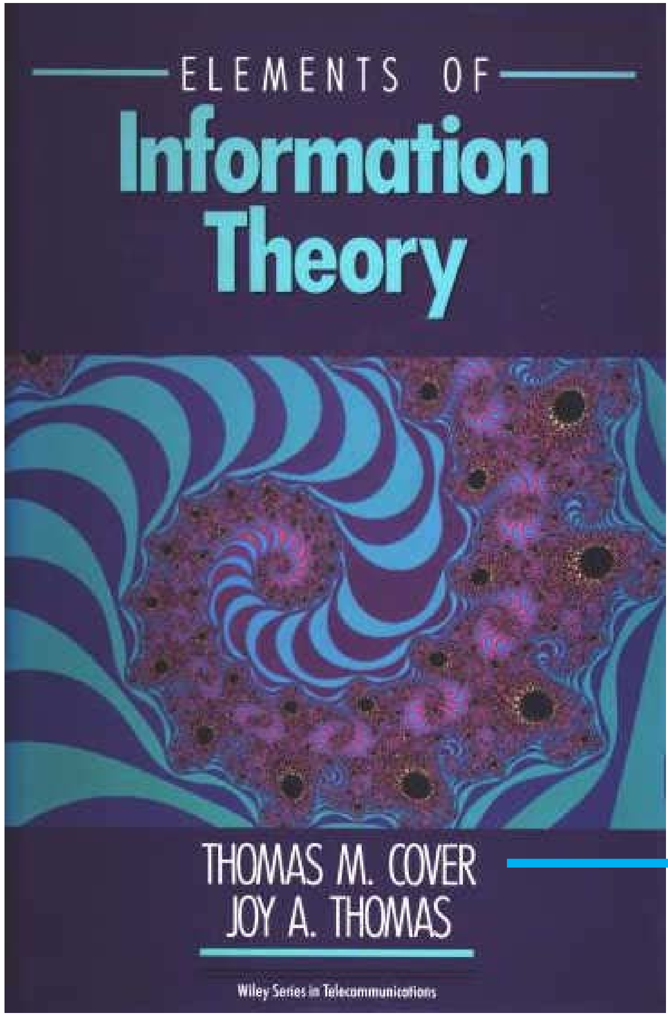
Office Hours:
BKD, 4th floor of Sirindhralai building

Monday	14:00-16:00
Thursday	10:30-11:30
Friday	15:00-16:00

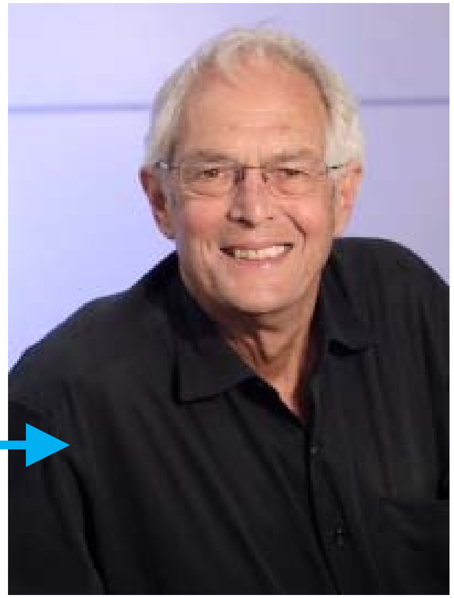
Elements of digital commu. sys.



Reference



- Elements of Information Theory
- 2006, 2nd Edition
- Chapters 2, 4 and 5



'the jewel in Stanford's crown'

One of the greatest information theorists since Claude Shannon (and the one most like Shannon in approach, clarity, and taste).

The ASCII Coded Character Set

Bit Number					*6*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*
					5	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*
					4	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*
				Hex	*1st*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
3	*2*	*1*	*0*	*2nd*									
						0	16	32	48	64	80	96	112
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	A		LF	SUB	*	:	J	Z	j	z
1	0	1	1	B		VT	ESC	+	;	K	[	k	{
1	1	0	0	C		FF	FS	,	<	L	\	l	|
1	1	0	1	D		CR	GS	-	=	M	]	m	}
1	1	1	0	E		SO	RS	.	>	N	^	n	~
1	1	1	1	F		SI	US	/	?	O	—	o	DEL

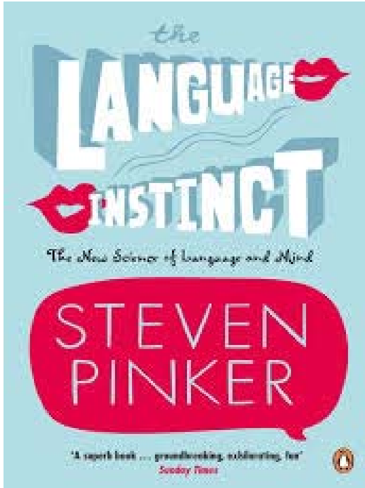
[The ARRL Handbook for Radio Communications 2013]

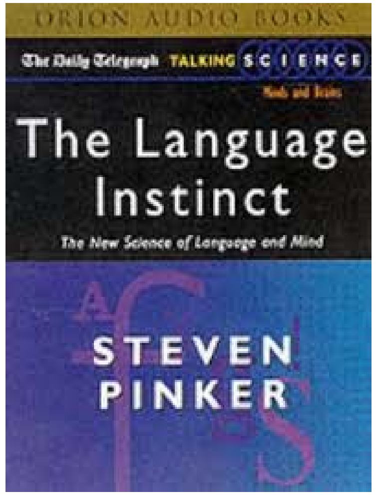
English Redundancy: Ex. 1

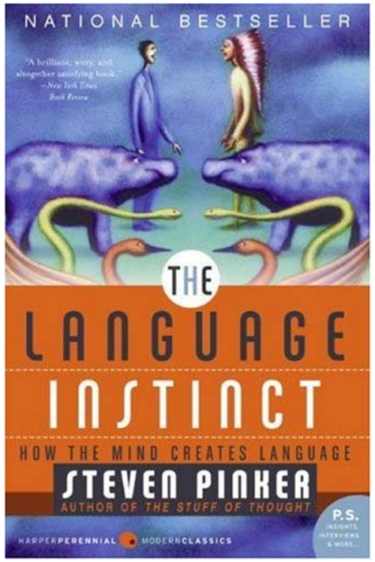
J-st tr- t- r--d th-s s-nt-nc-.

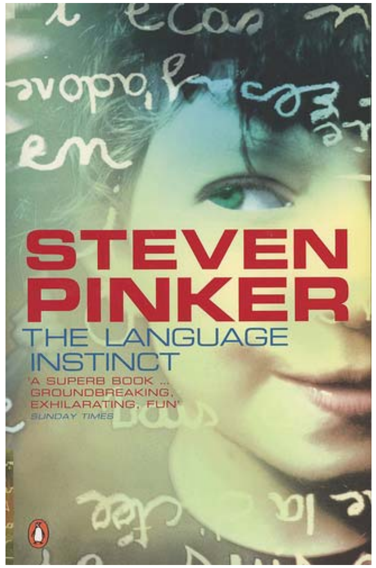
English Redundancy: Ex. 2

yxx cxn xndxrstxnd whxt x xm wrxtxng xvxn xf x rxplxcx xll thx vxwxls wxth xn 'x' (t gts lttl hrdr f y dn't vn kn whr th vwls r).









English Redundancy: Ex. 3

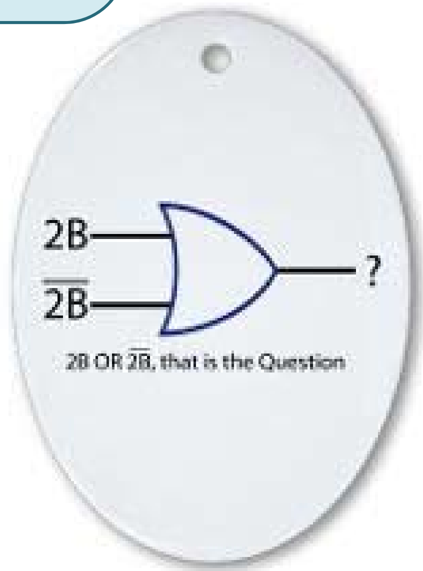
To be, or xxx xx xx,

xxxx xx xxx xxxxxxxxx

English Redundancy: Ex. 3

To be, or xxx xx xx,

xxxx xx xxx xxxxxxxxx



Ex. DMS (1)

$$\mathcal{S}_X = \{a, b, c, d, e\}$$

$$p_X(x) = \begin{cases} 1/5, & x \in \{a, b, c, d, e\} \\ 0, & \text{otherwise} \end{cases}$$

Information Source →

a	c	a	c	e	c	d	b	c	e
d	a	e	e	d	a	b	b	b	d
b	b	a	a	b	e	b	e	d	c
c	e	d	b	c	e	c	a	a	c
a	a	e	a	c	c	a	a	d	c
d	e	e	a	a	c	a	a	a	b
b	c	a	e	b	b	e	d	b	c
d	e	b	c	a	e	e	d	d	c
d	a	b	c	a	b	c	d	d	e
d	c	e	a	b	a	a	c	a	d

→

Approximately 20% are letter ‘a’s

[GenRV_Discrete_datasample_Ex1.m]

Ex. DMS (2)

$$\mathcal{S}_X = \{1,2,3,4\}$$

$$p_X(x) = \begin{cases} \frac{1}{2}, & x = 1, \\ \frac{1}{4}, & x = 2, \\ \frac{1}{8}, & x \in \{3,4\} \\ 0, & \text{otherwise} \end{cases}$$

Information Source →

2	1	1	2	1	4	1	1	1	1
1	1	4	1	1	2	4	2	2	1
3	1	1	2	3	2	4	1	2	4
2	1	1	2	1	1	3	3	1	1
1	3	4	1	4	1	1	2	4	1
4	1	4	1	2	2	1	4	2	1
4	1	1	1	1	2	1	4	2	4
2	1	1	1	2	1	2	1	3	2
2	1	1	1	1	1	1	2	3	2
2	1	1	2	1	4	2	1	2	1

Approximately 50% are number '1's

[GenRV_Discrete_datasample_Ex2.m]

DMS in MATLAB

```
clear all; close all;

S_X = [1 2 3 4]; p_X = [1/2 1/4 1/8 1/8]; n = 1e6;

SourceString = randsrc(1,n,[S_X;p_X]);
```

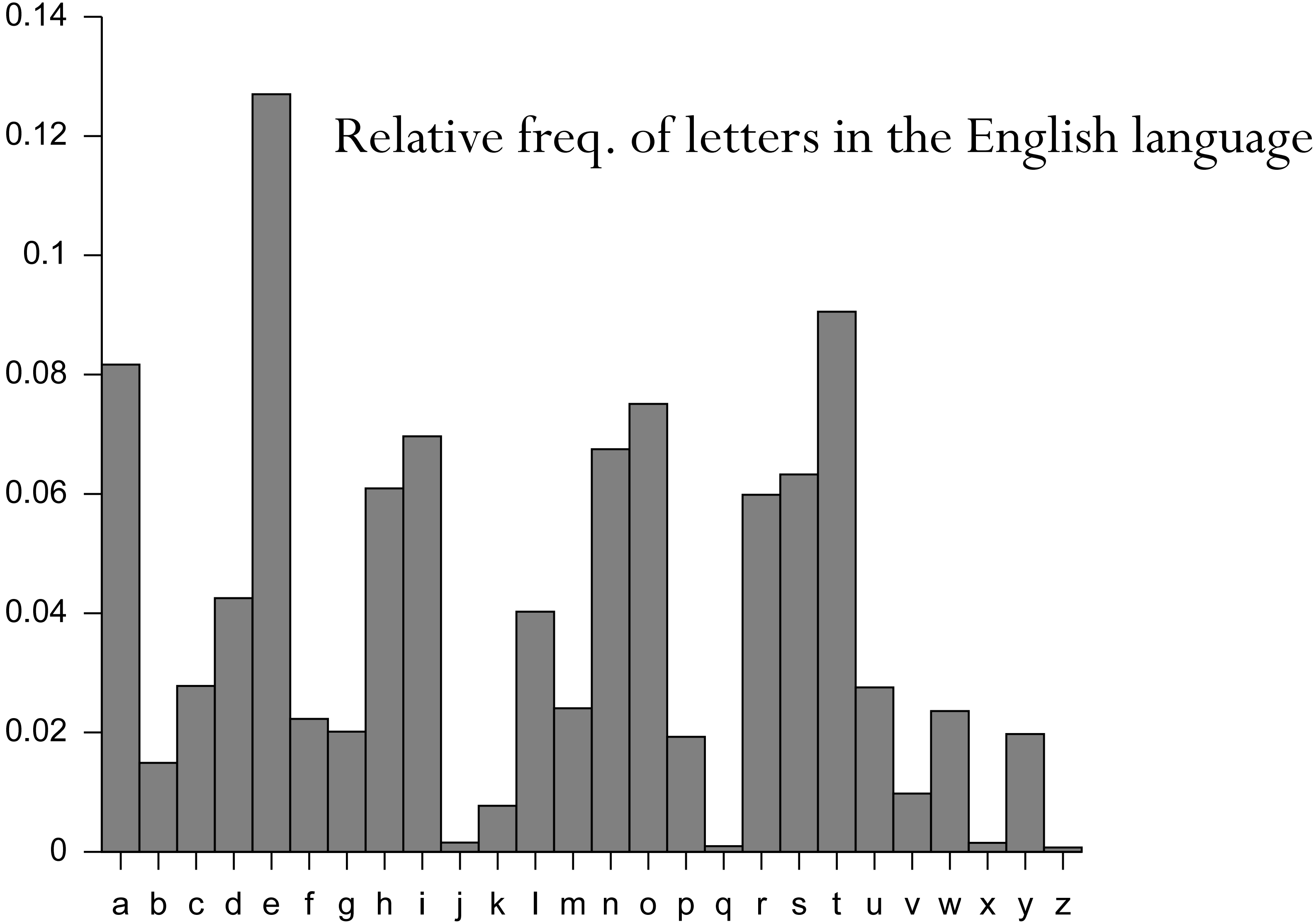
Alternatively, we can also use

```
SourceString = datasample(S_X,n,'Weights',p_X);
```

```
rf = hist(SourceString,S_X)/n; % Ref. Freq. calc.
stem(S_X,rf,'rx','LineWidth',2) % Plot Rel. Freq.
hold on
stem(S_X,p_X,'bo','LineWidth',2) % Plot pmf
xlim([min(S_X)-1,max(S_X)+1])
legend('Rel. freq. from sim.','pmf p_X(x)')
xlabel('x')
grid on
```

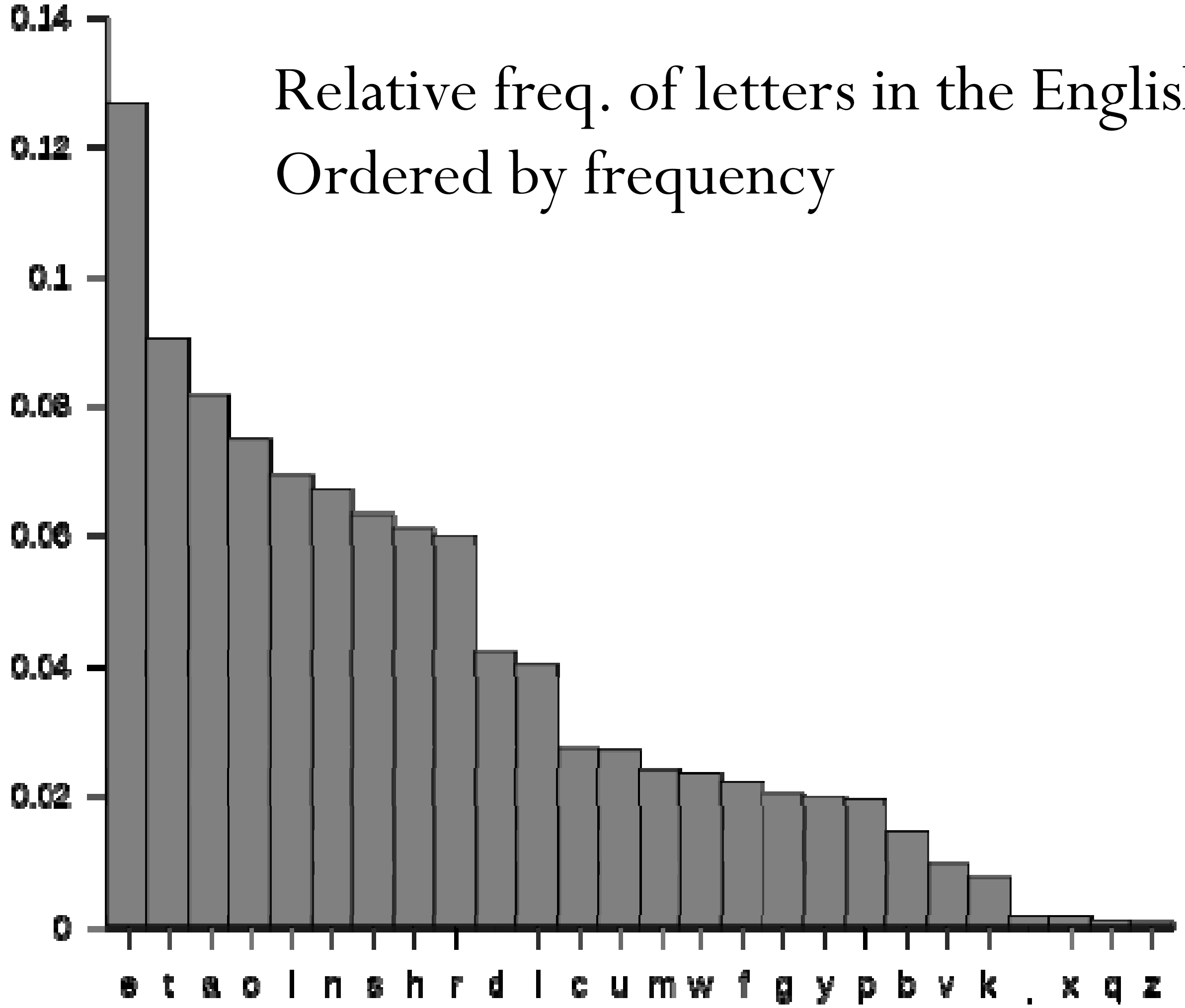
[GenRV_Discrete_datasample_Ex.m]

A more realistic example of pmf:



[http://en.wikipedia.org/wiki/Letter_frequency]

A more realistic example of pmf:



Relative freq. of letters in the English language
Ordered by frequency

[http://en.wikipedia.org/wiki/Letter_frequency]

Example: ASCII Encoder

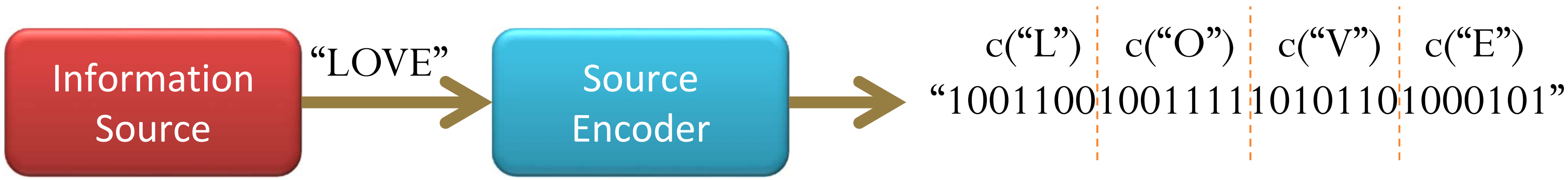
Codebook

Character x	Codeword c(x)
⋮	
E	1000101
⋮	
L	1001100
⋮	
O	1001111
⋮	
V	1010110
⋮	

MATLAB:

```
>> M = 'LOVE';
>> X = dec2bin(M,7);
>> X = reshape(X',1,numel(X))
X =
1001100100111110101101000101
```

Remark:
`numel(A) = prod(size(A))`
(the number of elements in matrix A)



A Byte (8 bits) vs. 7 bits

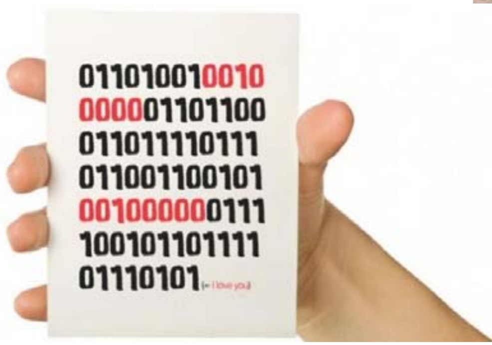
```
>> dec2bin('I Love ECS452',7)
ans =
1001001
0100000
1001100
1101111
1110110
1100101
0100000
1000101
1000011
1010011
0110100
0110101
0110010
```

```
>> dec2bin('I Love ECS452',8)
ans =
01001001
00100000
01001100
01101111
01110110
01100101
00100000
01000101
01000011
01010011
00110100
00110101
00110010
```

Geeky ways to express your love

```
>> dec2bin('I Love You',8)

ans =

01001001
00100000
01001100
01101111
01110110
01100101
00100000
01011001
01101111
01110101
```


```
>> dec2bin('i love you',8)

ans =

01101001
00100000
01101100
01101111
01110110
01100101
00100000
01111001
01101111
01110101
```



https://www.etsy.com/listing/91473057/binary-i-love-you-printable-for-your?ref=sr_gallery_9&ga_search_query=binary&ga_filters=holidays+-supplies+valentine&ga_search_type=all&ga_view_type=gallery
http://mentalfloss.com/article/29979/14-geeky-valentines-day-cards
https://www.etsy.com/listing/174002615/binary-love-geeky-romantic-pdf-cross?ref=sr_gallery_26&ga_search_query=binary&ga_filters=holidays+-supplies+valentine&ga_search_type=all&ga_view_type=gallery
https://www.etsy.com/listing/185919057/i-love-you-binary-925-silver-dog-tag-can?ref=sc_3&plkey=cdf3741cf5c63291bbc127f1fa7fb03e641daafd%3A185919057&ga_search_query=
&ga_filters=holidays+-supplies+valentine&ga_search_type=all&ga_view_type=gallery
http://www.cafepress.com/+binary-code+long_sleeve_tees

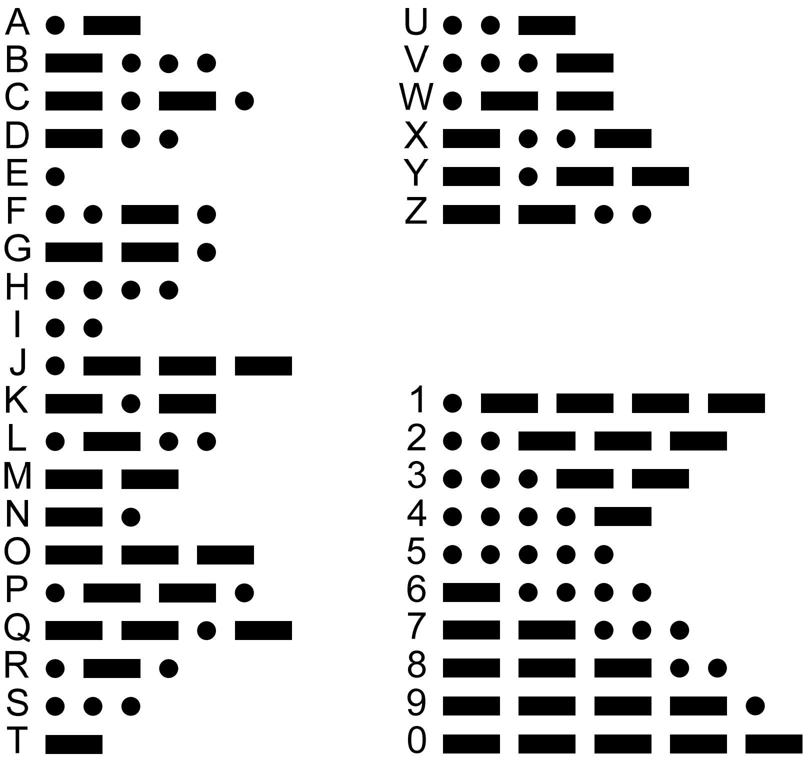
Source alphabet of size = 4

Morse code

(wired and wireless)

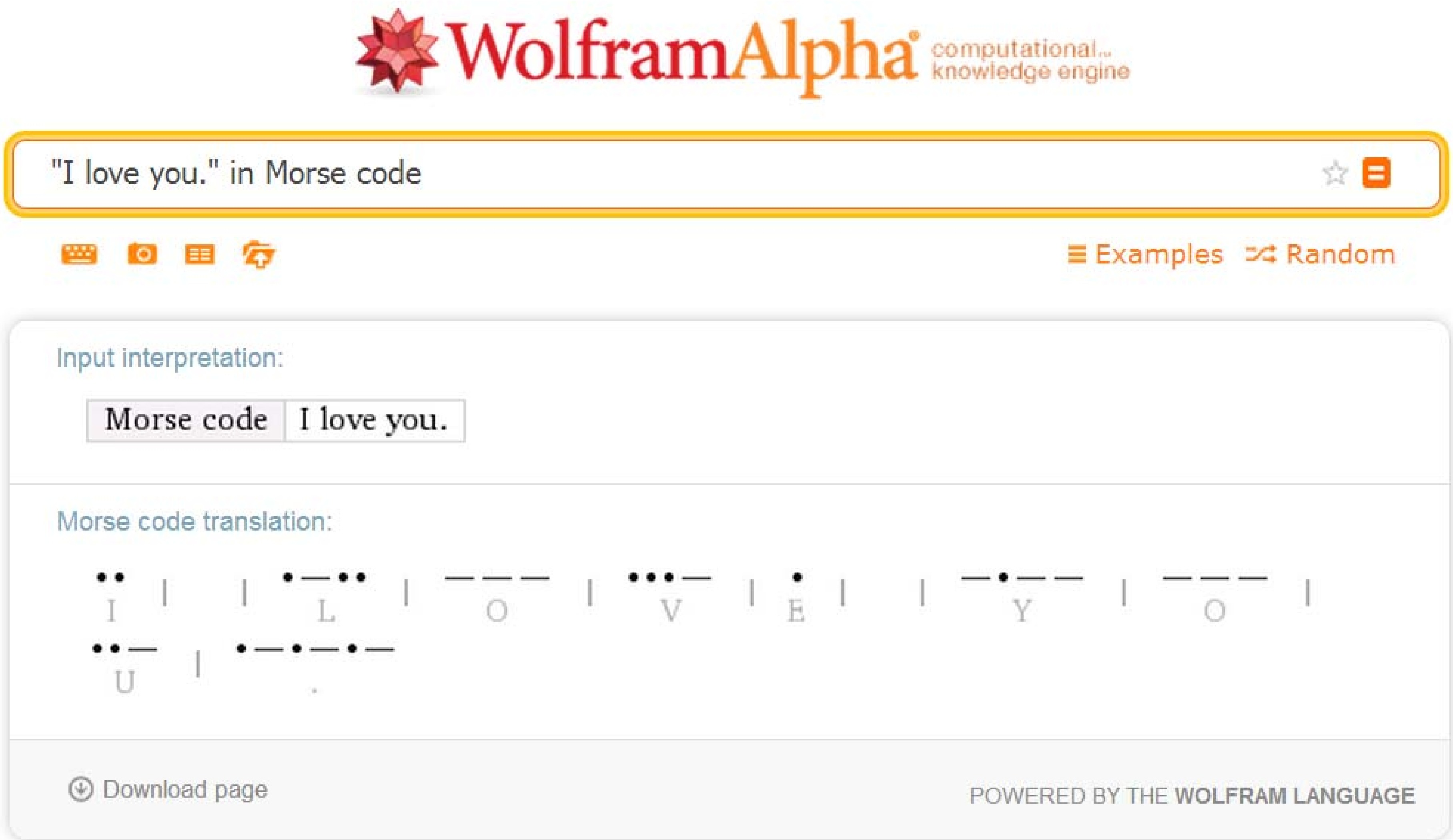
- **Telegraph network**
- Samuel **Morse**, 1838

- A sequence of on-off tones (or , lights, or clicks)



SAMUEL MORSE DICTATES A LETTER TO HIS SECRETARY.

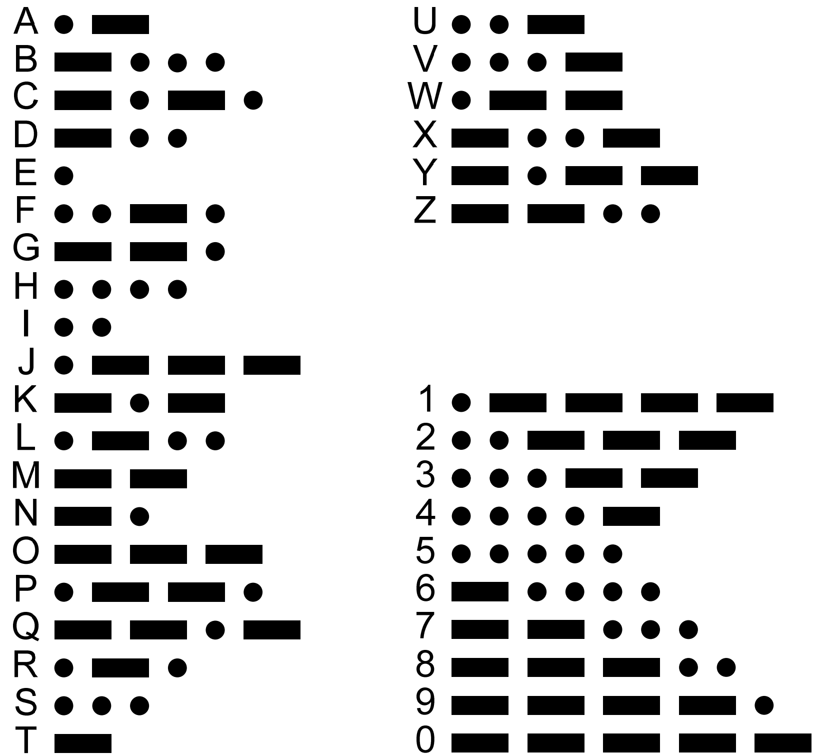
Example



[http://www.wolframalpha.com/input/?i=%22I+love+you.%22+in+Morse+code]

Morse code: Key Idea

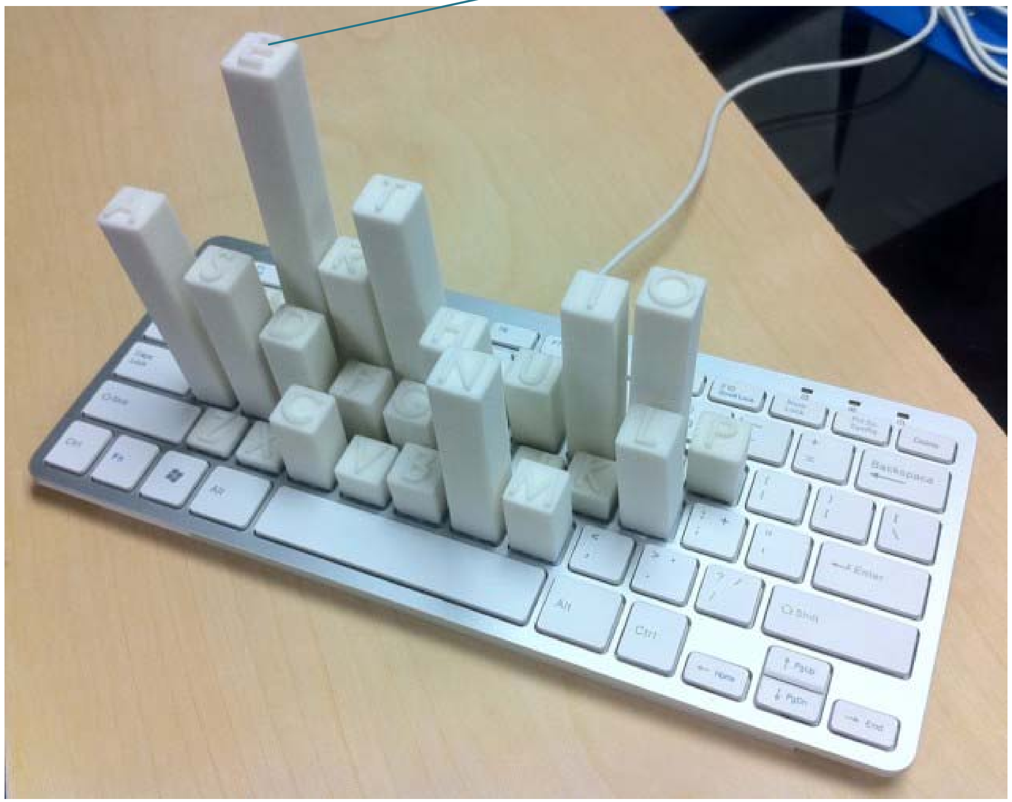
Frequently-used characters (e,t) are mapped to short codewords.



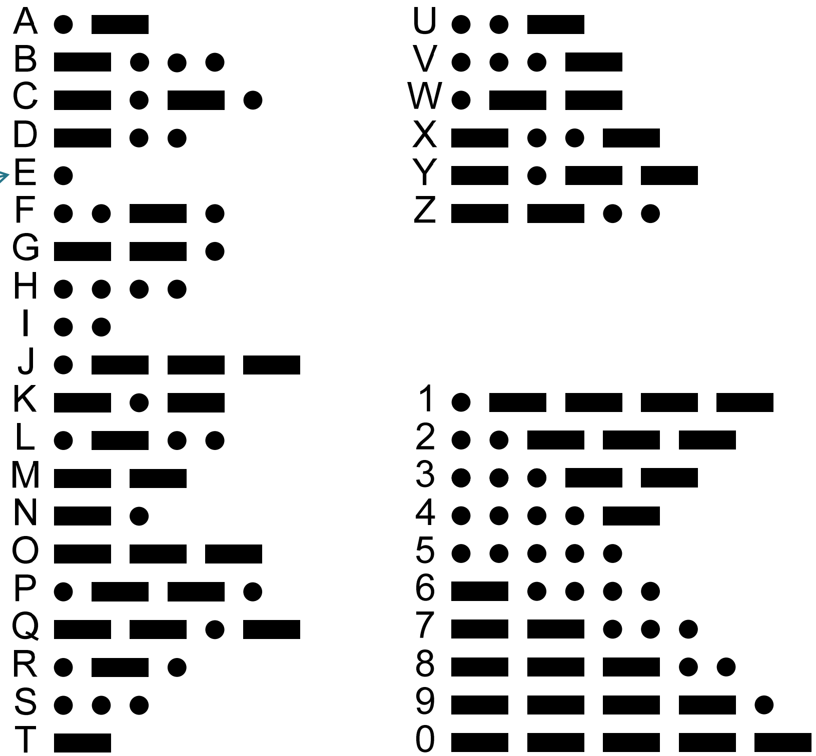
Basic form of compression.

Morse code: Key Idea

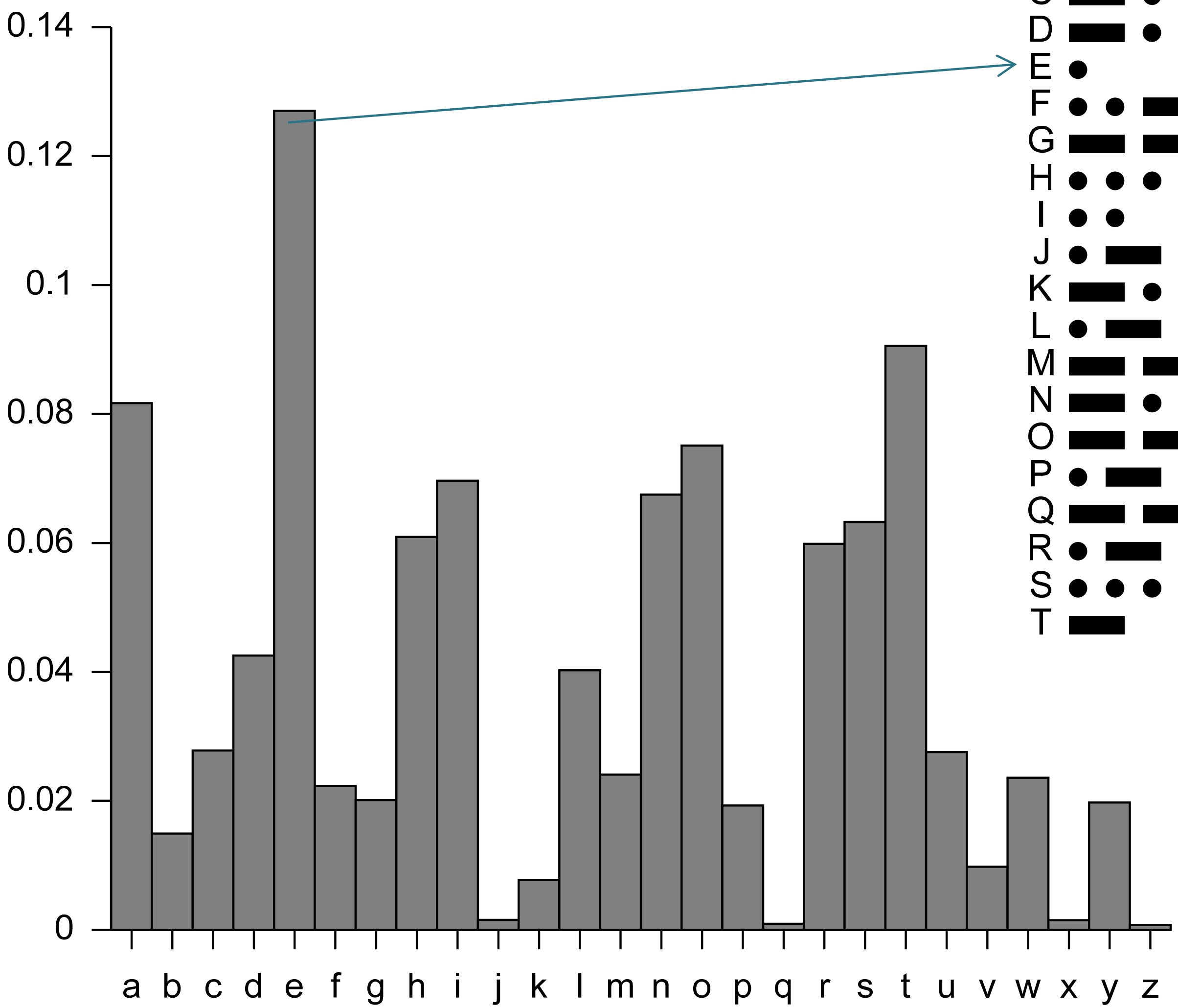
Frequently-used characters are mapped to short codewords.

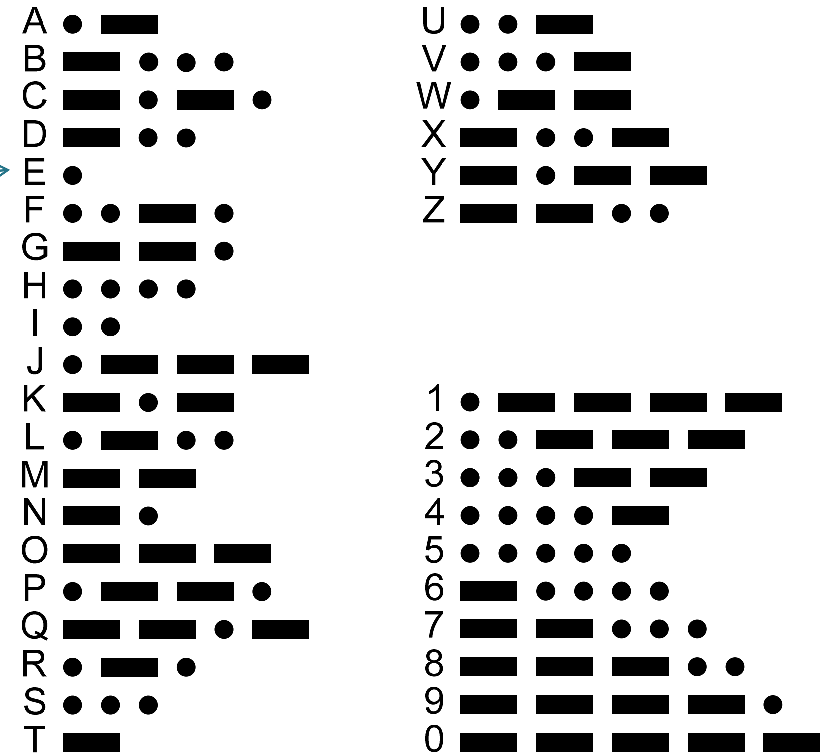


Relative frequencies of letters in the English language



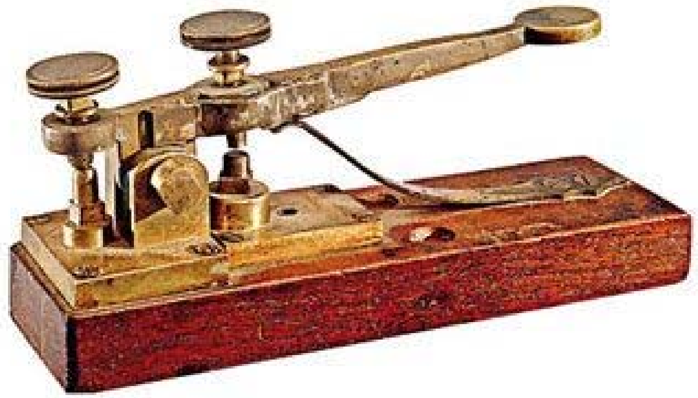
Morse code: Key Idea

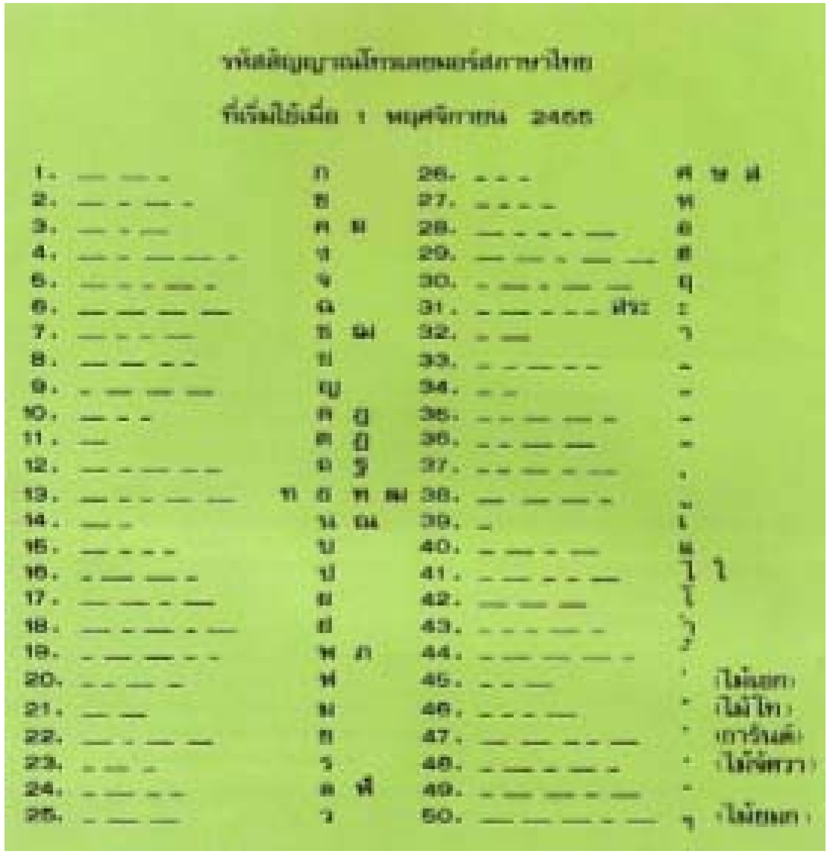




Frequently-used characters are mapped to short codewords.

รหัสมอร์สภาษาไทย

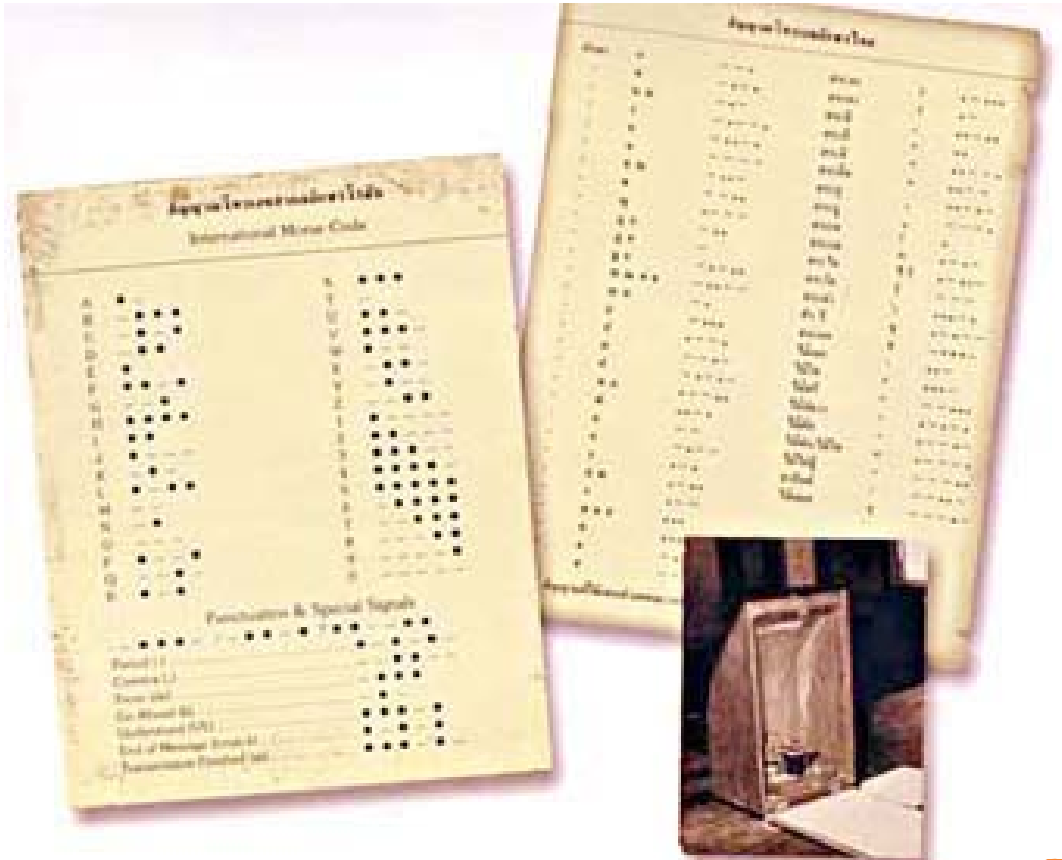




รหัสสัญญาณโทรเลขมอร์สภาษาไทย

ที่เริ่มใช้เมื่อ 1 พฤศจิกายน 2455

1.	– – .	ก	26.	. . .	ศ ษ ส
2.	– . – .	ข	27.	. . . .	ห
3.	– . –	ค ฆ	28.	– . . . –	อ
4.	– . – – .	ง	29.	– – . – –	ฮ
5.	– . . – .	จ	30.	. – . – –	ฤ
6.	– – – –	ฉ	31.	. – . . . สระ	ะ
7.	– . . –	ช ฌ	32.	. –	า
8.	– – . .	ซ	33.	. . – . .	ิ
9.	. – – –	ญ	34.	. .	ี
10.	– . .	ด ฎ	35.	. . – – .	ึ
11.	–	ต ฏ	36.	– – .	ื
12.	– . – . .	ถ ฐ	37.	. . – . –	ุ
13.	– . . – –	ฑ ท ธ ฒ	38.	– – – .	ู
14.	– .	น ณ	39.	.	เ
15.	– . . .	บ	40.	. – . –	แ
16.	. – – .	ป	41.	. – . . –	ไ ใ
17.	– – . –	ผ	42.	– – –	โ
18.	– . – . –	ฝ	43.	. . . – .	ำ
19.	. – – . .	พ ภ	44.	. – – . –	ั
20.	. . – .	ฟ	45.	. . –	่ (ไม้เอก)
21.	– –	ม	46.	. . . –	้ (ไม้โท)
22.	– . – –	ย	47.	– – . . –	์ (การันต์)
23.	. – .	ร	48.	. – . – .	๋ (ไม้จัตวา)
24.	. – . .	ล ฬ	49.	– – . . . –	๊
25.	. – –	ว	50.	– – – . –	ๆ (ไม้ยมก)

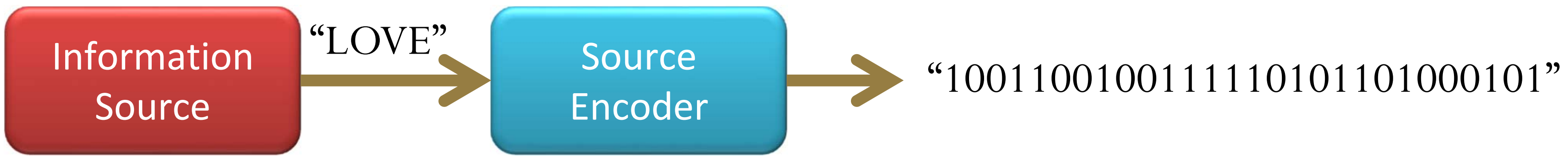


Example: ASCII Encoder

Character	Codeword
⋮	
E	1000101
⋮	
L	1001100
⋮	
O	1001111
⋮	
V	1010110
⋮	

MATLAB:

```
>> M = 'LOVE';
>> X = dec2bin(M,7);
>> X = reshape(X',1,numel(X))
X =
1001100100111110101101000101
```



Another Example of non-UD code

x	c(x)
A	1
B	011
C	01110
D	1110
E	10011

- Consider the string 011101110011.
- It can be interpreted as
  - CDB: 01110 1110 011
  - BABE: 011 1 011 10011

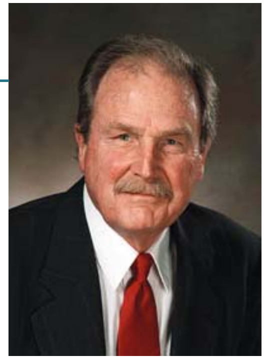
Summary

- A good code must be uniquely decodable (UD).
  - Difficult to check.
- A special family of codes called prefix(-free) code is always UD.
  - They are also instantaneous.
- Huffman's recipe
  - Repeatedly combine the two least-likely symbols
  - Automatically give prefix code
- For a given source's pmf, Huffman codes are optimal among all UD codes for that source.

Prof. Robert Mario Fano (MIT)
Shannon Award (1976)

Shannon–Fano coding

- Proposed in Shannon's "A Mathematical Theory of Communication" in 1948
- The method was attributed to Fano, who later published it as a technical report.
- Should not be confused with
  - Shannon coding, the coding method used to prove Shannon's noiseless coding theorem, or with
  - Shannon–Fano–Elias coding (also known as Elias coding), the precursor to arithmetic coding.



David Huffman (1925–1999)

Huffman Code

- MIT, 1951
- Information theory class taught by Professor Fano.
- Huffman and his classmates were given the choice of
    - a term paper on the problem of finding the most efficient binary code.

  or
    - a final exam.
- Huffman, unable to prove any codes were the most efficient, was about to give up and start studying for the final when he hit upon the idea of using a frequency-sorted binary tree and quickly proved this method the most efficient.
- Huffman avoided the major flaw of the suboptimal Shannon-Fano coding by building the tree from the bottom up instead of from the top down.

Claude E. Shannon Award

Claude E. **Shannon** (1972)
David S. Slepian (1974)
Robert M. **Fano** (1976)
Peter Elias (1977)
Mark S. Pinsker (1978)
Jacob Wolfowitz (1979)
W. Wesley Peterson (1981)
Irving S. Reed (1982)
Robert G. Gallager (1983)
Solomon W. Golomb (1985)
William L. Root (1986)
James L. Massey (1988)
Thomas M. Cover (1990)
Andrew J. Viterbi (1991)

Elwyn R. Berlekamp (1993)
Aaron D. Wyner (1994)
G. David Forney, Jr. (1995)
Imre Csiszár (1996)
Jacob Ziv (1997)
Neil J. A. Sloane (1998)
Tadao Kasami (1999)
Thomas Kailath (2000)
Jack Keil Wolf (2001)
Toby **Berger** (2002) →
Lloyd R. Welch (2003)
Robert J. McEliece (2004)
Richard Blahut (2005)
Rudolf Ahlswede (2006)

Sergio Verdu (2007)
Robert M. Gray (2008)
Jorma Rissanen (2009)
Te Sun Han (2010)
Shlomo Shamai (Shitz) (2011)
Abbas El Gamal (2012)
Katalin Marton (2013)
János Körner (2014)
Arthur Robert Calderbank (2015)

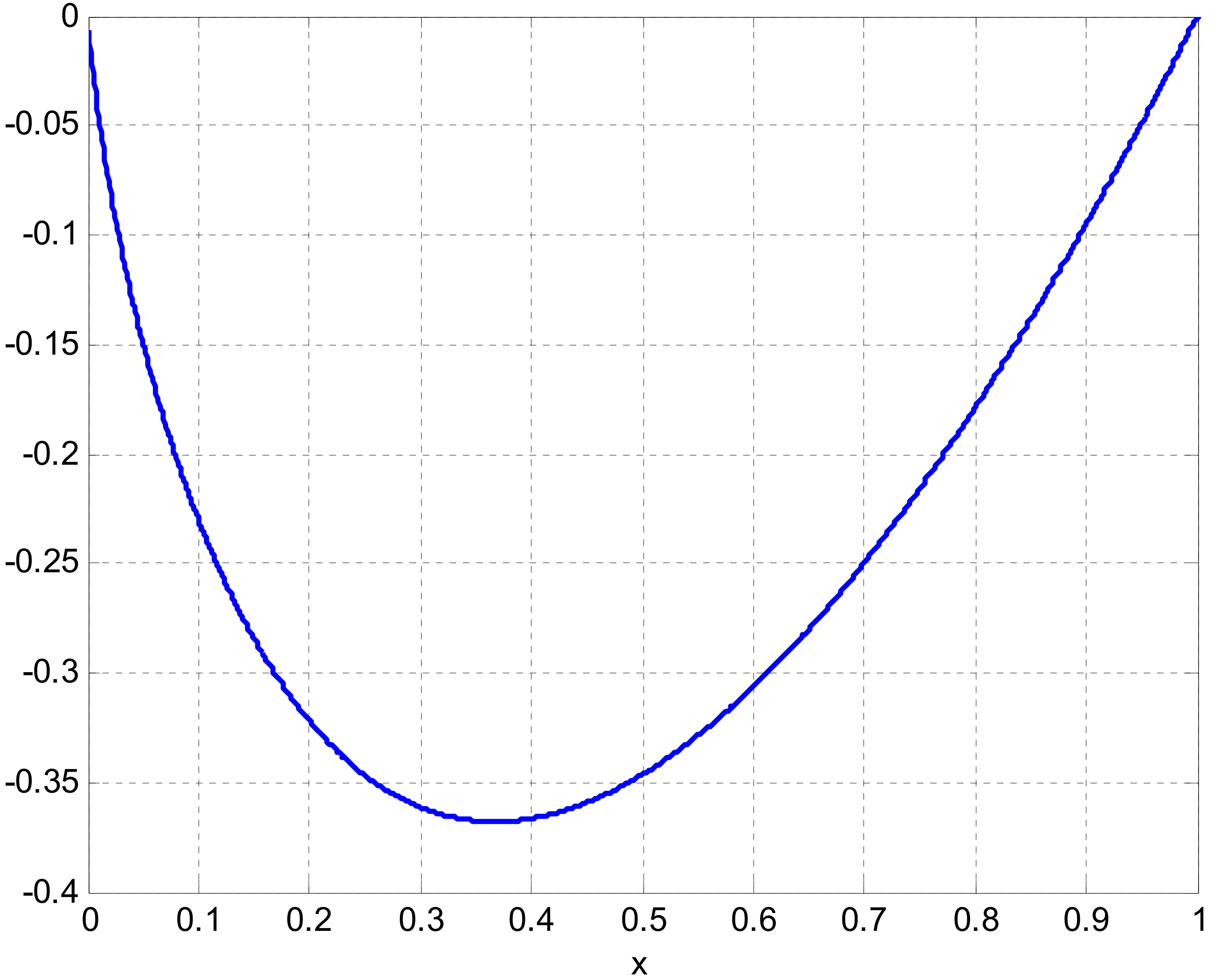
IEEE Richard W. Hamming Medal

1988 - Richard W. **Hamming**
1989 - Irving S. Reed
1990 - Dennis M. Ritchie and Kenneth L. Thompson
1991 - Elwyn R. Berlekamp
1992 - Lotfi A. Zadeh
1993 - Jorma J. Rissanen
1994 - Gottfried Ungerboeck
1995 - Jacob Ziv
1996 - Mark S. Pinsker
1997 - Thomas M. Cover
1998 - David D. Clark
1999 - David A. **Huffman**
2000 - Solomon W. Golomb
2001 - A. G. Fraser
2002 - Peter Elias
2003 - Claude Berrou and Alain Glavieux
2004 - Jack K. Wolf
2005 - Neil J.A. Sloane
2006 - Vladimir I. Levenshtein
2007 - Abraham Lempel
2008 - Sergio Verdú
2009 - Peter Franaszek
2010 - Whitfield Diffie, Martin Hellman and Ralph Merkle
2011 - Toby **Berger**
2012 - Michael Luby, Amin Shokrollahi
2013 - Arthur Robert Calderbank
2014 - Thomas Richardson and Rüdiger L. Urbanke

"For contributions to Information Theory, including **source coding** and its applications."

[http://www.cvaieee.org/html/toby_berger.html]

$x\ln(x)$



Ex. Huffman Coding in MATLAB

[Ex. 2.31]

Observe that MATLAB automatically give the **expected length** of the codewords

```
pX = [0.5 0.25 0.125 0.125];        % pmf of X
SX = [1:length(pX)];                % Source Alphabet
[dict,EL] = huffmandict(SX,pX);     % Create codebook
```

```
%% Pretty print the codebook.
codebook = dict;
for i = 1:length(codebook)
    codebook{i,2} = num2str(codebook{i,2});
end
codebook

%% Try to encode some random source string
n = 5; % Number of source symbols to be generated
sourceString = randsrc(1,10,[SX; pX]) % Create data using pX
encodedString = huffmanenco(sourceString,dict) % Encode the data
```

[Huffman_Demo_Ex1]

Ex. Huffman Coding in MATLAB

```
codebook =

    [1]    '0'
    [2]    '1  0'
    [3]    '1  1  1'
    [4]    '1  1  0'


sourceString =

     1     4     4     1     3     1     1     4     3     4


encodedString =

     0  1  1  0  1  1  0  0  1  1  1  0  0  1  1  0  1  1  1  1  1  0
```


[Huffman_Demo_Ex1]

Ex. Huffman Coding in MATLAB

[Ex. 2.32]

```
pX = [0.4 0.3 0.1 0.1 0.06 0.04]; % pmf of X
SX = [1:length(pX)];              % Source Alphabet
[dict,EL] = huffmandict(SX,pX);   % Create codebook

%% Pretty print the codebook.
codebook = dict;
for i = 1:length(codebook)
    codebook{i,2} = num2str(codebook{i,2});
end
codebook

EL
```

```
>> Huffman_Demo_Ex2

codebook =

    [1]    '1'
    [2]    '0  1'
    [3]    '0  0  0  0'
    [4]    '0  0  1'
    [5]    '0  0  0  1  0'
    [6]    '0  0  0  1  1'


EL =

    2.2000
```

The codewords can be different from our answers found earlier.

The expected length is the same.

[Huffman_Demo_Ex2]

Ex. Huffman Coding in MATLAB

[Exercise]

```
pX = [1/8, 5/24, 7/24, 3/8]; % pmf of X
SX = [1:length(pX)];              % Source Alphabet
[dict,EL] = huffmandict(SX,pX);   % Create codebook

%% Pretty print the codebook.
codebook = dict;
for i = 1:length(codebook)
    codebook{i,2} = num2str(codebook{i,2});
end
codebook

EL
```

```
>> -pX*(log2(pX)).'
ans =
     1.8956
```

```
codebook =
    [1]    '0  0  1'
    [2]    '0  0  0'
    [3]    '0  1'
    [4]    '1'
EL =
    1.9583
```

Kronecker Product

- An operation on two matrices of arbitrary size
- Named after German mathematician Leopold Kronecker.
- If **A** is an *m*-by-*n* matrix and **B** is a *p*-by-*q* matrix, then the **Kronecker product** **A**⊗**B** is the *mp*-by-*nq* matrix

Use `kron(A,B)` in MATLAB.

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

- Example

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1\cdot 0 & 1\cdot 5 & 2\cdot 0 & 2\cdot 5 \\ 1\cdot 6 & 1\cdot 7 & 2\cdot 6 & 2\cdot 7 \\ 3\cdot 0 & 3\cdot 5 & 4\cdot 0 & 4\cdot 5 \\ 3\cdot 6 & 3\cdot 7 & 4\cdot 6 & 4\cdot 7 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}.$$

Kronecker Product

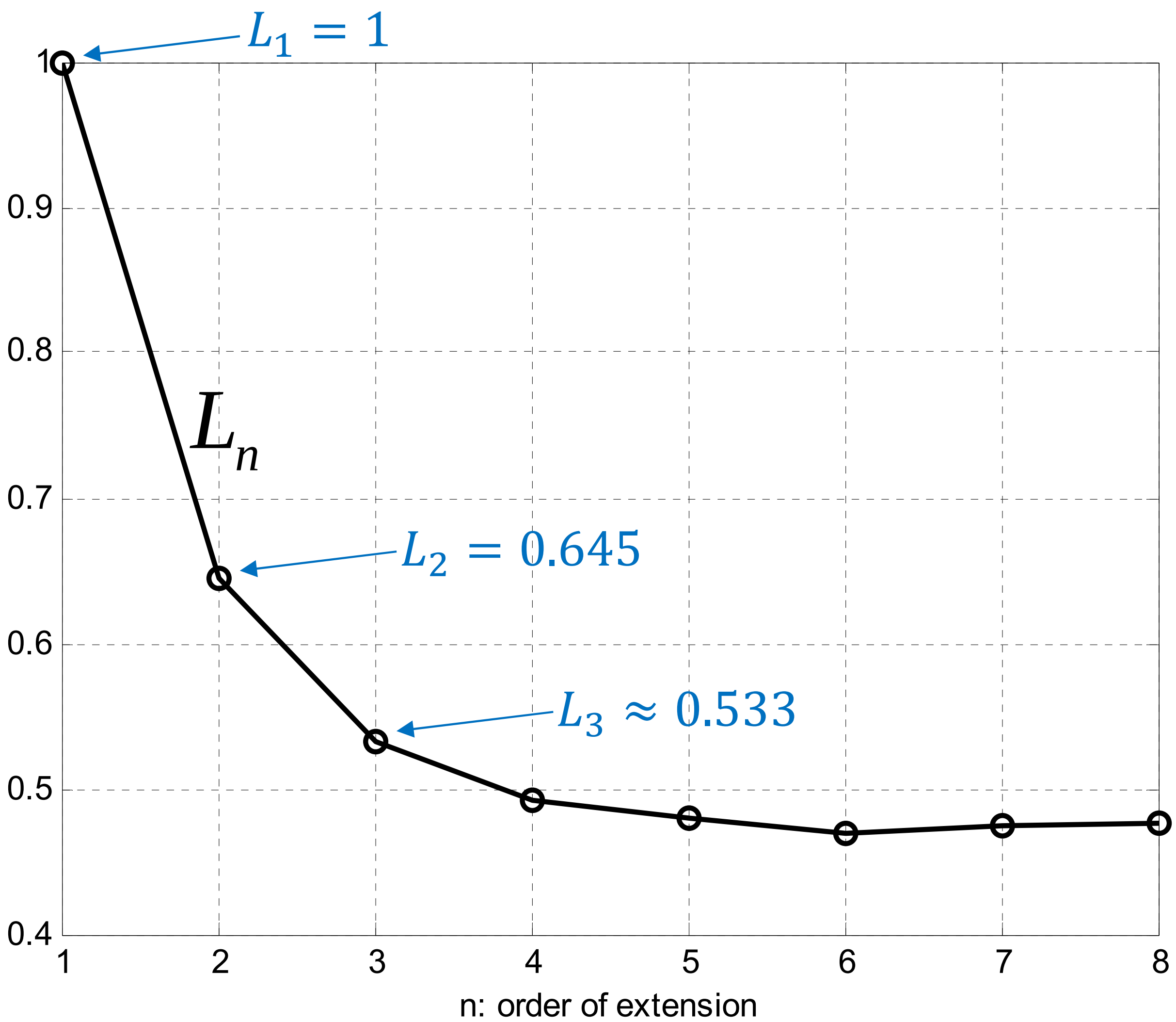
```
>> p = [0.9 0.1]
p =
    0.9000    0.1000
>> p2 = kron(p,p)
p2 =
    0.8100    0.0900    0.0900    0.0100
>> p3 = kron(p2,p)
p3 =
  Columns 1 through 7
    0.7290    0.0810    0.0810    0.0090    0.0810    0.0090    0.0090
  Column 8
    0.0010
```

[Ex.2.40]

Huffman Coding: Source Extension

$$X_k \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p)$$

$$p = 0.1$$

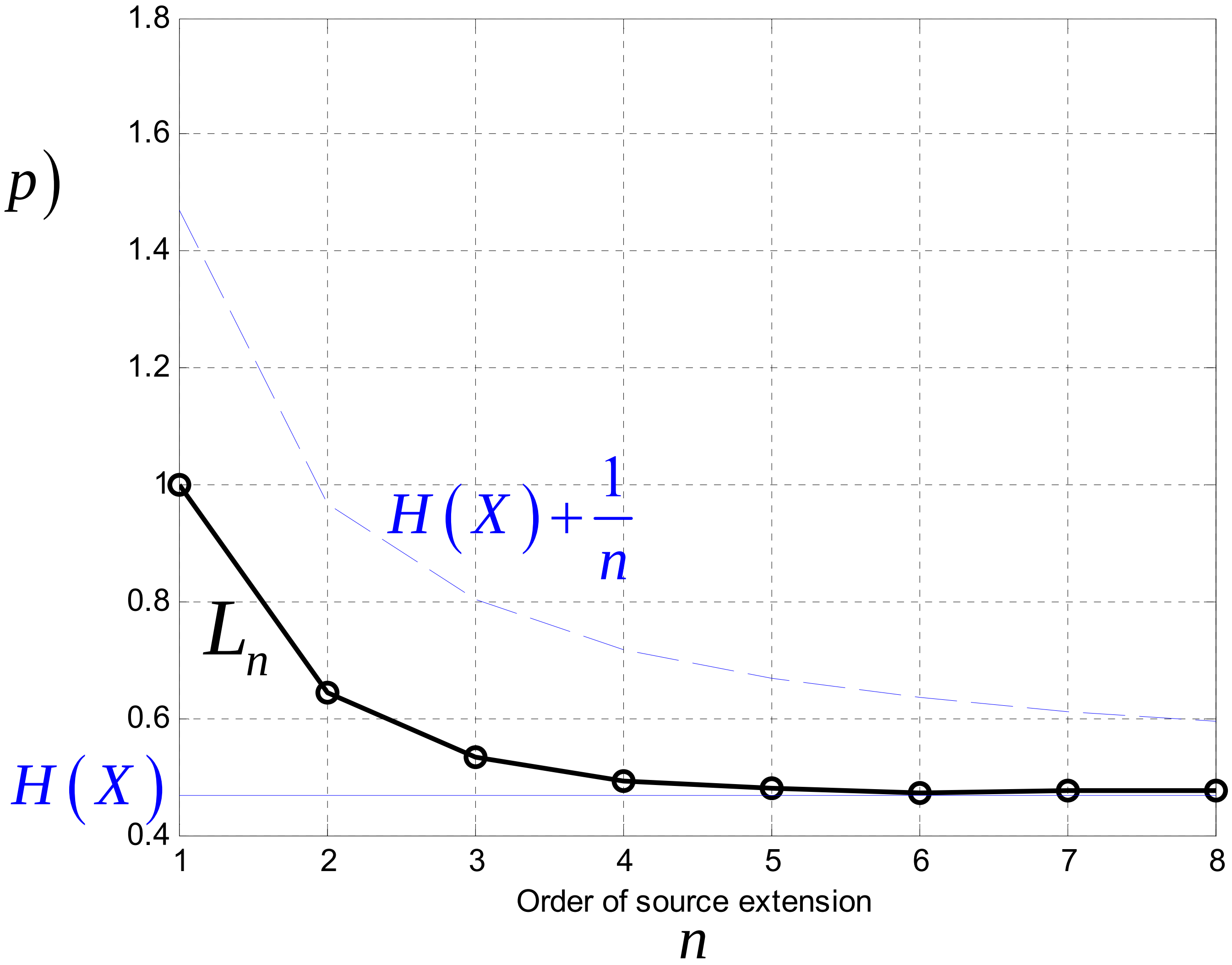


[Ex.2.40]

Huffman Coding: Source Extension

$$X_k \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p)$$

$$p = 0.1$$



What to do when the pmf is unknown?

- One may assume uniform pmf
  - Inefficient if the actual pmf is not uniform.
- Better Solution: **universal** lossless data compression algorithms
  - Universal source coding
  - **Lempel-Ziv** algorithm is one popular example.