Űrkommunikáció
Space Communication
2023/4.

Source coding, Type I:

Encoding (mapping) the source symbols **without considering the Entropy** of the source.

- This is not exactly a compressing source coding, just a (for example binary) representation of the source symbols.
- **No a-prior knowledge of statistical properties** (first or higher order PDF) of the source are needed.

Encode **fixed length k source words to fixed length l code words**:

n-ary source symbol set: $X = \{x_1, x_2, \cdots, x_n\}$

r-ary code symbol set: $U = \{u_1, u_2, \cdots, u_r\}$

Must be ***<u>explicit</u>***:

$$n^k \leq r^l$$
$$k \cdot ld\, n \leq l \cdot ld\, r$$

Definition: Coding rate ***R***

$$R = \frac{l}{k} \geq \frac{ld\, n}{ld\, r} = \frac{H_0(X)}{ld\, r} \xrightarrow[r=2\ (binary)]{} H_0(X)\ \left[\frac{Shannon}{source\ symbol}\right]$$

Shannon's I. Theorem – Source Coding Theorem: If we apply a coding rate

$$R \geq H(X) + \varepsilon;\ \varepsilon > 0,$$

Then the probability of decoding error can be made arbitrary small: $P_e \to 0$

Source coding, Type I, Example:

Encode **fixed length k source words to fixed length l code words**:

n=5 source symbol set: $X = \{A, B, C, D, E\}$

Entropy of the source (no knowledge of the statistics):

$$H_0(X) = ld\ n = ld\ 5 \cong 2{,}3219 \left[\frac{bit}{symbol}\right]$$

r=2 binary source code symbol set: $U = \{0{,}1\}$

source word length k	**code word length l**	**Coding rate *R***
$k = 1$ (symbol by symbol):	$5 \le 2^l \to l = 3$	$R = \frac{l}{k} = 3 \left[\frac{bit}{symbol}\right]$

Source extension

$k = 2$ (pair of symbols):	$5^2 = 25 \le 2^l \to l = 5$	$R = \frac{5}{2} = 2{,}5 \left[\frac{bit}{symbol}\right]$
$k = 3$ (triple of symbols):	$5^3 = 125 \le 2^l \to l = 7$	$R = \frac{7}{3} = 2{,}333\dot{3} \left[\frac{bit}{symbol}\right]$

However not convergent:

$k = 10$	$10 \cdot ld\ 5 \cong 23{,}219 \le l \to l = 24$	$R = \frac{24}{10} = 2{,}4 \left[\frac{bit}{symbol}\right]$

ASCII (*American Standard Code for Information Interchange)*

Source symbol set size n=256, Code symbol set size r=2 (binary)
Encode source symbols by symbol (k=1) to code words of length l=8 binary digits.

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

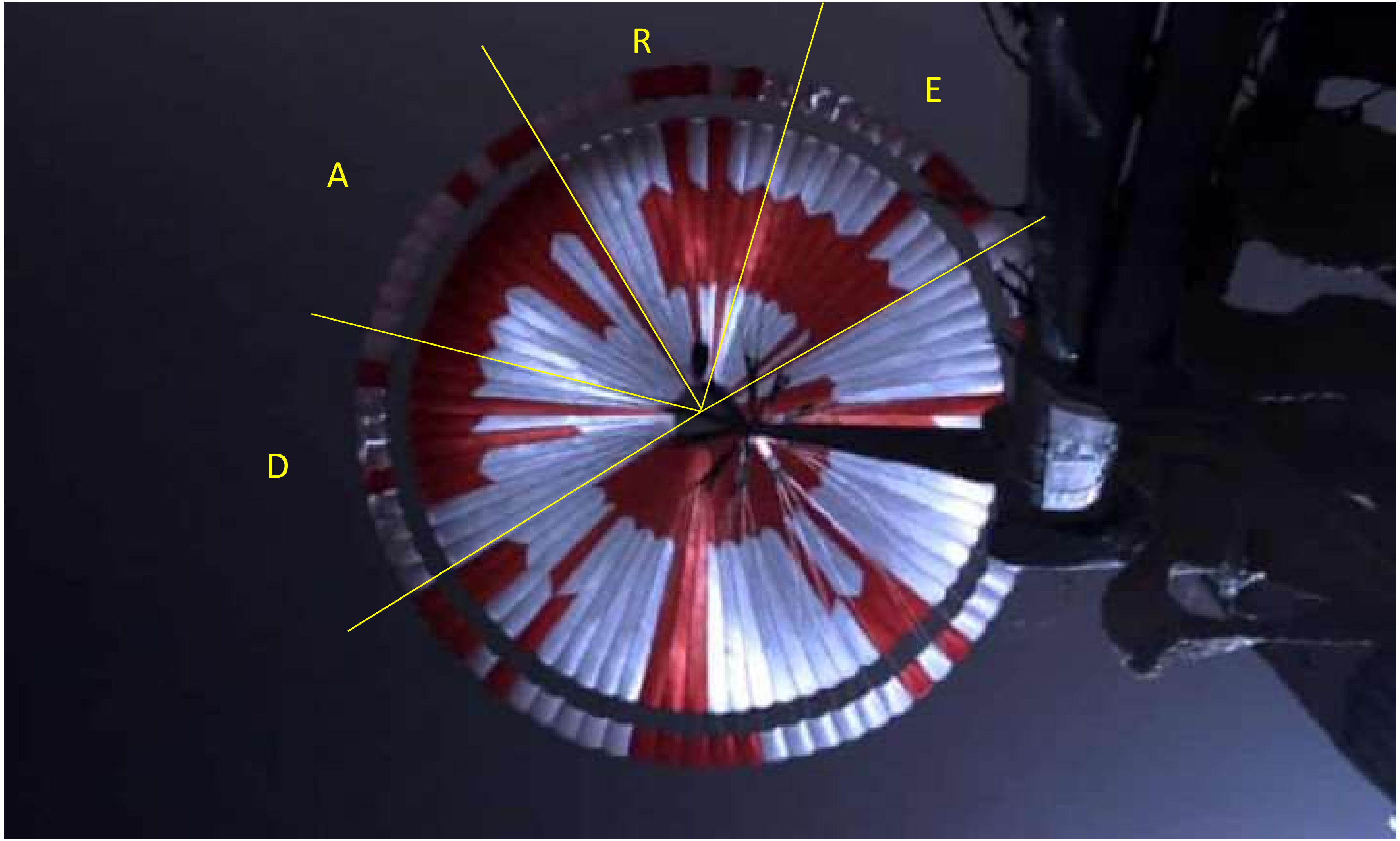
Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Canopy of Perseverance Mars rover

ASCII encoded message: „DARE MIGHTY THINGS”

E.g.: 7 white, 1 red and 2 white slice =0000000100=4; 4+64=68=D



Source coding, ***Entropy Coding***

Encoding the source **with considering the Entropy is also called *Entropy Coding*** (Type II):

- Encode **fixed length k source words to variable length l code words**
- **a-priori knowledge of statistical properties** (first – or even higher – order PDF) of the source are needed.

Shannon's method: Code word length should be reciprocally proportional to the corresponding source word probability.

$$l(\bar{u}) \sim \frac{1}{p(\bar{x})}$$

By an ***r***-ary code symbol set: $l(\bar{u}) = \left\lceil log_r \frac{1}{p(\bar{x})} \right\rceil$, where $\lceil z \rceil$ is the smallest integer greater than or equal to z. *Notation:* $l(\bar{u}_{\,i}) = l_i$ length of the i-th code word.

Symbol by symbol ($k = 1$) **encoding into binary code words** (r=2)

n-ary source symbol set: $X = \{x_1, x_2, \cdots, x_n\}$ with PDF: $p(X) = \{p(x_1), p(x_2), \cdots, p(x_n)\}$

$$ld \frac{1}{p(x_i)} \leq l_i < ld \frac{1}{p(x_i)} + 1 \quad \forall i$$

$$\sum_{i=1}^{n} p(x_i) \cdot ld \frac{1}{p(x_i)} \leq L(X) = \sum_{i=1}^{n} p(x_i) \cdot l_i < \sum_{i=1}^{n} p(x_i) \cdot ld \frac{1}{p(x_i)} + \sum_{i=1}^{n} p(x_i)$$

Shannon's I. Theorem – Source Coding Theorem: $\boldsymbol{H(X) \leq L(X) < H(X) + 1}$

Source coding, ***Entropy Coding***

Definition: ***Code efficiency***

$$h = \frac{H(X)}{L(X)} \leq 1 \; (or\ in\ percent \leq 100\%)$$

Special case: $\forall\, p(x_i) = \mathbf{2^{-l_i}} \rightarrow \boldsymbol{h = 100\%}$

Source Extension for improving efficiency: **encoding a block of source symbols** (k>1) into a code word:

$$ld \frac{1}{p(x_1, x_2, \dots, x_k)} = ld \frac{1}{p(\bar{x})} \leq l(\bar{u}) < ld \frac{1}{p(\bar{x})} + 1 \quad \forall \bar{x}$$

$$\sum_{\bar{x}} p(\bar{x}) \cdot ld \frac{1}{p(\bar{x})} \leq L(X_1, \dots, X_k) = L(\bar{X}) = \sum_{\bar{x}} p(\bar{x}) \cdot l(\bar{u}) < \sum_{\bar{x}} p(\bar{x}) \cdot ld \frac{1}{p(\bar{x})} + \sum_{\bar{x}} p(\bar{x})$$

$$H(X_1, \dots, X_k) \leq L(X_1, \dots, X_k) < H(X_1, \dots, X_k) + 1$$

$$H(\bar{X}) \leq L(\bar{X}) < H(\bar{X}) + 1$$

DMS: $$H(\bar{X}) = k \cdot H(X) \leq L(\bar{X}) < k \cdot H(X) + 1$$

$$\rightarrow \boldsymbol{H(X)} \leq \frac{L(\bar{X})}{k} = \boldsymbol{L(X)} < \boldsymbol{H(X)} + \frac{\mathbf{1}}{\boldsymbol{k}} \rightarrow \frac{k \cdot L(X) - 1}{k \cdot L(X)} < h \leq 1$$

k-th order stationary:

$$H_k(X) = \frac{1}{k} H(\bar{X}) \leq L(X) < H_k(X) + 1/k$$

Strict stationary:

$$H_\infty(X) \leq L(X) < H_\infty(X) + \varepsilon$$

Shannon's I. Theorem – Source Coding Theorem

Shannon’s algorithm

Consider a discrete random variable X whit n possible values:

$$X = \{x_1, x_2, \cdots, x_n\}$$

With a Probability Density Function (PDF) ordered the events by **decreasing** probability:

$$p(X) = \{p(x_1) \geq p(x_2) \geq \cdots \geq p(x_n)\}$$

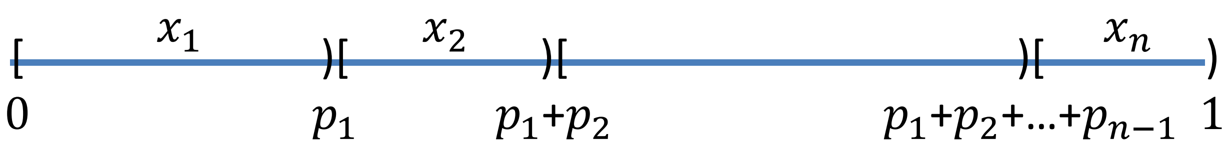
- **Determine code word lengths first**: $l(\bar{u}_{\,i}) = l_i = \left\lceil log_r \frac{1}{p(x_i)} \right\rceil$,
- **Determine code words** in order, choosing the **lexicographically** first word of the correct length that maintains the prefix condition.

Or Alternatively

- **Determine code words** in order using the **cumulative probability method**

✓ Consider the partitioning of the zero to one interval according the event probabilities into subintervals. Each subinterval closed from the left side (lower limit) and open from the right side (upper limit) corresponds to an event. At the end we have the **cumulated probabilities** 1=$\sum_{i=1}^{n} p(x_i)$

✓ Obtain the first l_i r-ary digits of the lower limit value regarded as r-ary rational number (base-r numeral system) as corresponding code word.



Shannon's algorithm, example

$$X = \{A, B, C, D, E\}, \quad p(X) = \{\text{p(A)=0.385, p(B)=0.179, p(C)=0.154, p(D)=0.154, p(E)=0.128}\},$$

$$H(X) \cong \mathbf{2,185356} \left[\frac{Shannon}{symbol}\right]$$

Determine binary (r=2) code word set.

Symbols	A	B	C	D	E
Probabilities $p(x_i)$	0.385	0.179	0.154	0.154	0.128
Word lengths $l_i = \left\lceil ld \frac{1}{p(x_i)} \right\rceil$	2	3	3	3	3
Code words (lexicographically)	**00**	**010**	**011**	**100**	**101**
Cumulative probabilities	0.000	0.385	0.564	0.718	0.872
...in binary	0.00000	0.01100	0.10010	0.10110	0.11011
Code words (cumulative probability)	**00**	**011**	**100**	**101**	**110**

Prefix redundant: $\sum_{i=1}^{5} 2^{-l_i}$ =0.75, Average length: $\sum_{i=1}^{5} p(x_i) \cdot l_i$ =**2,615** $\left[\frac{bit}{symbol}\right]$, $h \cong 83{,}57\%$

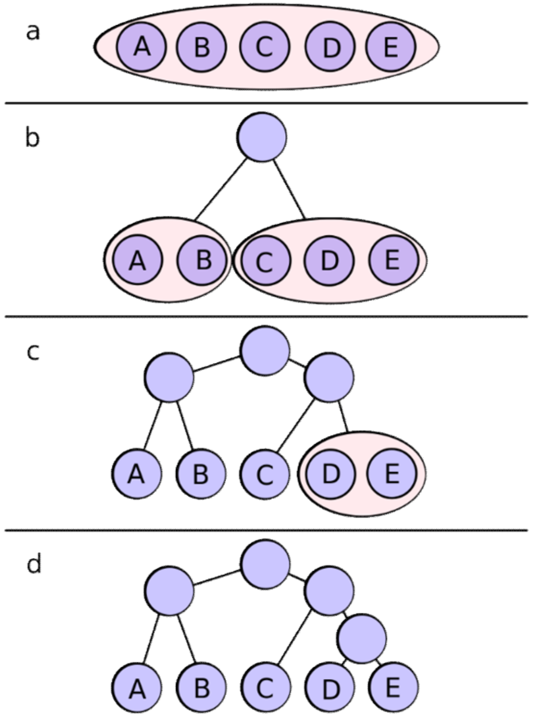
Reduce the longest Code words	**00**	**011**	**100**	**101**	**110**

Prefix redundant: $\sum_{i=1}^{5} 2^{-l_i}$ =0.875, Average length: $\sum_{i=1}^{5} p(x_i) \cdot l_i$ =**2,436** $\left[\frac{bit}{symbol}\right]$, $h \cong 89{,}71\%$

Reduce the longest Code words	**00**	**01**	**100**	**101**	**110**

Prefix complete: $\sum_{i=1}^{5} 2^{-l_i}$ =1, Average length: $\sum_{i=1}^{5} p(x_i) \cdot l_i$ =**2,308** $\left[\frac{bit}{symbol}\right]$, $h \cong 94{,}69\%$

Fano's algorithm, Shannon-Fano code



Build a binary tree starting **from the root** of the tree.
The Shannon–Fano binary **tree construction algorithm**:

- ✓ Divide the list into two parts, with the total frequency counts of the left part being as close to the total of the right as possible.
- ✓ The left part of the list is assigned a binary digit, and the right part is assigned the other binary digit.

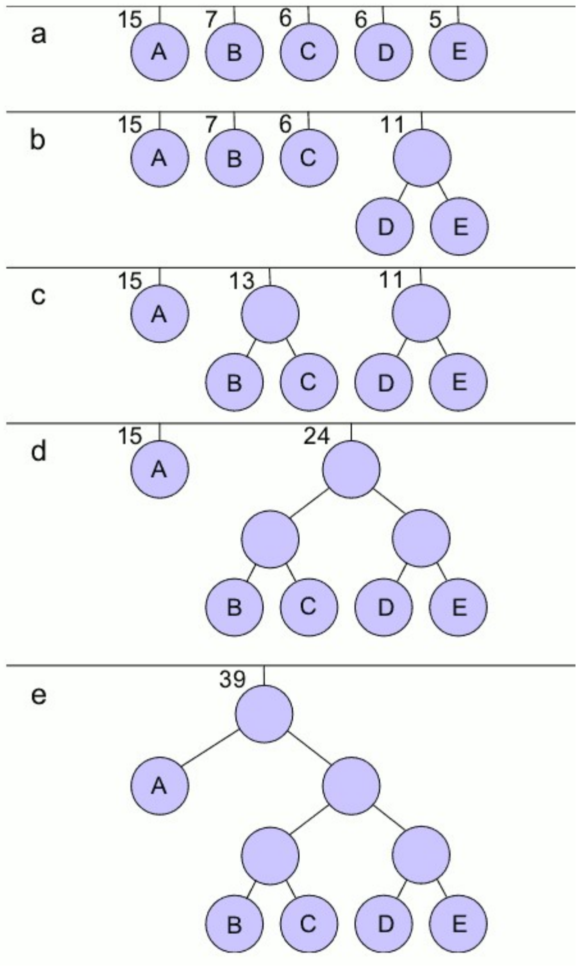
Recursively apply the steps to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree.

Example (cont. of the previous):

Symbols	A	B	C	D	E
Probabilities $p(x_i)$	0.385	0.179	0.154	0.154	0.128
b) First division	0			1	
c) Second division	0	1	0	1	
d) Third division				0	1
Code words	**00**	**01**	**10**	**110**	**111**

Prefix complete: $\sum_{i=1}^{5} \mathbf{2}^{-l_i} = 1$, Average length: $\sum_{i=1}^{5} p(x_i) \cdot l_i = \mathbf{2,282} \left[\frac{bit}{symbol}\right]$, $h \cong 95{,}76\%$

Huffman code



While Fano's Shannon–Fano **tree is created** by dividing from the root to the leaves, the **Huffman algorithm** works in the opposite direction, **merging from the leaves to the root**.
Create a leaf node for each symbol and add it to a priority queue, using its probability (or frequency of occurrence) as the priority.
While there is more than one node in the queue:

- ✓ Remove the two nodes of lowest probability from the queue
- ✓ Prepend 0 and 1 respectively to any code already assigned to these nodes
- ✓ Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
- ✓ Add the new node to the queue.

The remaining node is the root node and the tree is complete.

Example (cont. of the previous):

Symbols	A	B	C	D	E
Count (total 39)	15	7	6	6	5
Probabilities $p(x_i)$	0.385	0.179	0.154	0.154	0.128
Codewords	**0**	**100**	**101**	**110**	**111**

Prefix complete: $\sum_{\boldsymbol{i=1}}^{\mathbf{5}} \mathbf{2}^{-\boldsymbol{l_i}}$ =1, Average length: $\sum_{i=1}^{5} p(x_i) \cdot l_i = \mathbf{2,23} \left[\frac{bit}{symbol}\right]$, $h \cong 98\%$

Huffman code, another example

a-priori $p(X) = \{p(x_1) = 0.4, p(x_2) = 0.3, p(x_3) = 0.2, p(x_4) = 0.1\}$

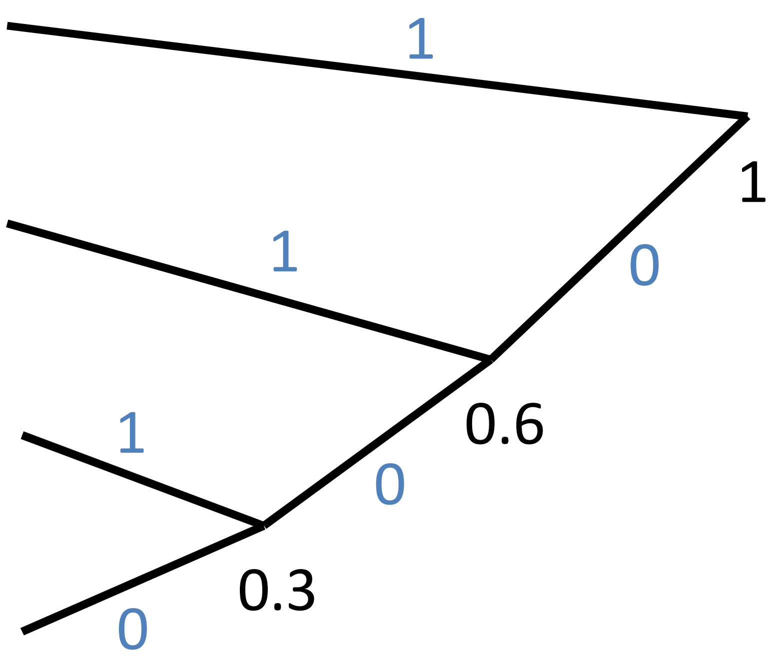
Code words:

x_1 0.4 — 1

x_2 0.3 — 01

x_3 0.2 — 001

x_4 0.1 — 000



$$\boldsymbol{H(X)} \cong \mathbf{1,846439} \left[\frac{Shannon}{symbol}\right]$$

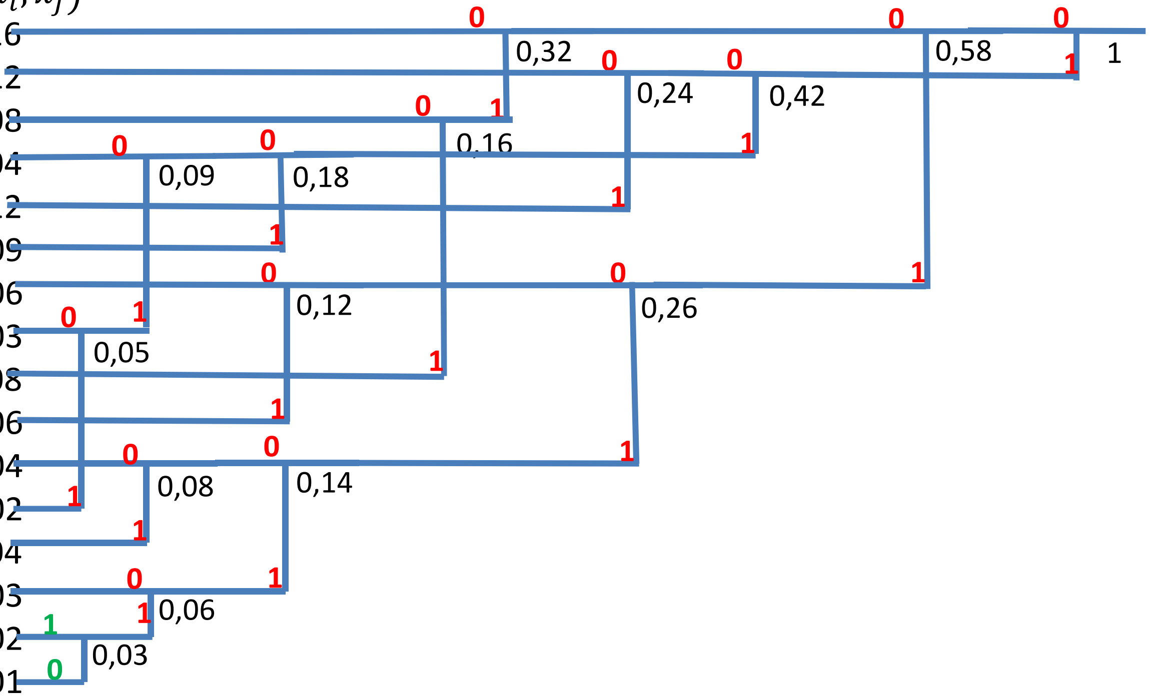
$$\boldsymbol{L(X)} = \sum_{i=1}^{4} p(x_i) \cdot l_i = 0{,}4 \cdot 1 + 0{,}3 \cdot 2 + 0{,}2 \cdot 3 + 0{,}1 \cdot 3 = \mathbf{1,9} \left[\frac{bit}{symbol}\right]$$

Efficiency $h = \frac{H(X)}{L(X)} \cong 97{,}18\%$, could be improved by source extension

Huffman code, Source extension, DMS example

Continue previous example for DMS by encoding pair of symbols: $p(x_i, x_j) = p(x_i) \cdot p(x_j)$

Codewords	XX	$p(x_i, x_j)$
000	x_1x_1	0,16
100	x_1x_2	0,12
0010	x_1x_3	0,08
1100	x_1x_4	0,04
101	x_2x_1	0,12
111	x_2x_2	0,09
0100	x_2x_3	0,06
11010	x_2x_4	0,03
0011	x_3x_1	0,08
0101	x_3x_2	0,06
01100	x_3x_3	0,04
11011	x_3x_4	0,02
01101	x_4x_1	0,04
01110	x_4x_2	0,03
011111	x_4x_3	0,02
011110	x_4x_4	0,01



$$L(XX) = \sum_{i=1}^{4}\sum_{j=1}^{4} p(x_i, x_j) \cdot l_{ij} = 0{,}49 \cdot 3 + 0{,}32 \cdot 4 + 0{,}16 \cdot 5 + 0{,}03 \cdot 6 = \mathbf{3{,}73} \left[\frac{bit}{2\ symbols}\right]$$

$$\boldsymbol{H(X)} \cong \mathbf{1{,}846439} \left[\frac{\boldsymbol{bit}}{\boldsymbol{symbol}}\right], \quad \boldsymbol{L(X)} = \frac{\boldsymbol{L(XX)}}{\mathbf{2}} = \mathbf{1{,}865} \left[\frac{\boldsymbol{bit}}{\boldsymbol{symbol}}\right], \quad \textbf{Efficiency } \boldsymbol{h} = \frac{\boldsymbol{H(X)}}{\boldsymbol{L(X)}} \cong \mathbf{99\%},$$

Kullback Leibler Distance, Kullback Leibler Divergence, Information Divergence, Relative Entropy

Consider a discrete random variable $X = \{x_1, x_2, x_3, x_4\}$

With the true Probability Density Function (PDF):

$$p(X) = \left\{p(x_1) = \frac{1}{2}, p(x_2) = 1/4, p(x_3) = 1/8, p(x_4) = 1/8\right\}$$

$$\boldsymbol{H(X) = 1,75} \left[\frac{\boldsymbol{bit}}{\boldsymbol{symbol}}\right]$$

And with our (not exact) assumed PDF:

$$q(X) = \left\{q(x_1) = \frac{1}{2}, q(x_2) = 1/8, q(x_3) = 1/4, p(x_4) = 1/8\right\}$$

Definition **Kullback Leibler Distance:**

$$D\big(p(X) \,||\, q(X)\big) = \sum_X p(x) \cdot ld \frac{p(x)}{q(x)} = \sum_X q(x) \cdot ld \frac{q(x)}{p(x)} = D\big(q(X) \,||\, p(X)\big)$$

X	p(x)	Code
x_1	½	0
x_2	¼	10
x_3	1/8	110
x_4	1/8	111

X	q(x)	Code
x_1	½	0
x_2	1/8	100
x_3	1/4	11
x_4	1/8	101

$\boldsymbol{L(X) = 1,75} \left[\frac{\boldsymbol{bit}}{\boldsymbol{symbol}}\right]$

$\boldsymbol{L_q(X) = \sum_{i=1}^{4} p(x_i) \cdot l_i}$ =**1,875** $\left[\frac{bit}{symbol}\right]$