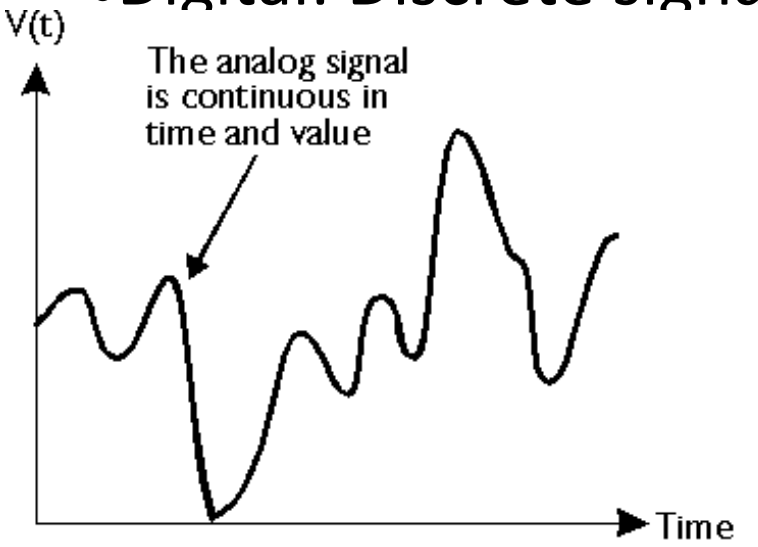
Data Representation

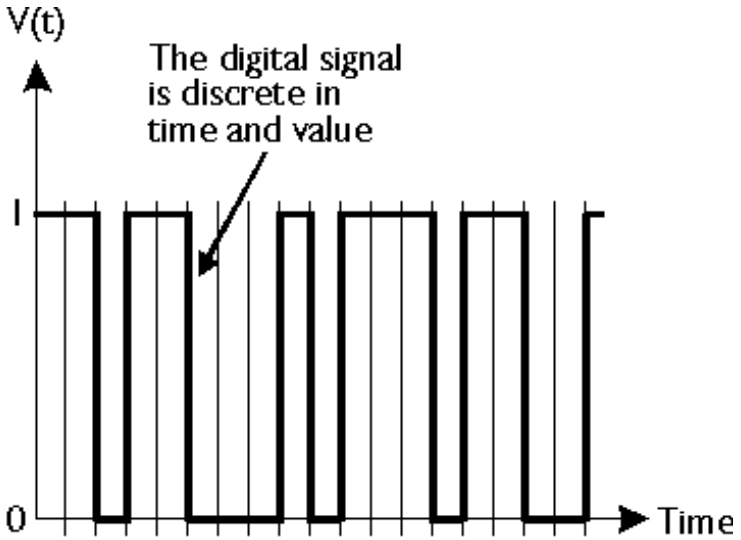
Data Representation

- Types of data:
  - Numbers
  - **Text**
  - Audio
  - Images & Graphics
  - Video

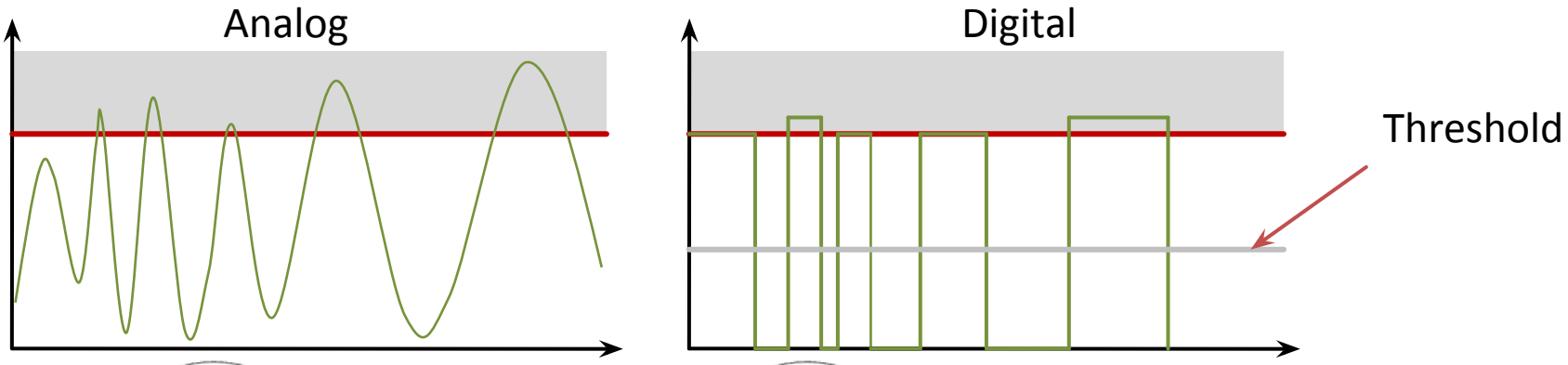
Analog vs Digital data

- How is data represented?
  - What is a signal?
    - Transmission of data
- Analog vs Digital
  - Analog: Continuous signal
  - Digital: Discrete signal





Analog vs Digital data



Analog

1. More accurate, infinite resolution

1. Easily affected by noise
2. Degrades over time -Data Loss
3. Hard to reproduce

Digital

1. Easy to code, decipher and maintain
2. Less expensive

1. Loss of Information due to quantization

Representing Text

- Document: Paragraphs, sentences, words
  - All made up of *characters*
- English language has 26 letters
  - 52 if you consider upper and lower case
  - Punctuation characters
  - Space
- Character sets: ASCII and Unicode

ASCII Character Set

Lower Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00		32	20		64	40	@	96	60	`
1	01	☺	33	21	!	65	41	A	97	61	a
2	02	☻	34	22	"	66	42	B	98	62	b
3	03	♥	35	23	#	67	43	C	99	63	c
4	04	♦	36	24	$	68	44	D	100	64	d
5	05	♣	37	25	%	69	45	E	101	65	e
6	06	♠	38	26	&	70	46	F	102	66	f
7	07	•	39	27	'	71	47	G	103	67	g
8	08	◘	40	28	(	72	48	H	104	68	h
9	09	○	41	29	)	73	49	I	105	69	i
10	0A	◙	42	2A	*	74	4A	J	106	6A	j
11	0B	♂	43	2B	+	75	4B	K	107	6B	k
12	0C	♀	44	2C	,	76	4C	L	108	6C	l
13	0D	♪	45	2D	-	77	4D	K	109	6D	m
14	0E	♫	46	2E	.	78	4E	N	110	6E	n
15	0F	☼	47	2F	/	79	4F	O	111	6F	o
16	10	►	48	30	0	80	50	P	112	70	p
17	11	◄	49	31	1	81	51	Q	113	71	q
18	12	↕	50	32	2	82	52	R	114	72	r
19	13	‼	51	33	3	83	53	S	115	73	s
20	14	¶	52	34	4	84	54	T	116	74	t
21	15	§	53	35	5	85	55	U	117	75	u
22	16	▬	54	36	6	86	56	V	118	76	v

Upper Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷

ASCII Character Set

256 characters – 8 bits = 1 byte

ASCII: Character *a*

--> Dec: 97 --> Binary: 01100001

Unicode Character Set

2^{16}: 65000 characters

ASCII is a subset of Unicode

Unicode Character Set

Why Unicode?

Þa ẏðan ᵹeꞇacniað þiꞅne deopan cꞃæꝼꞇ, and . . .

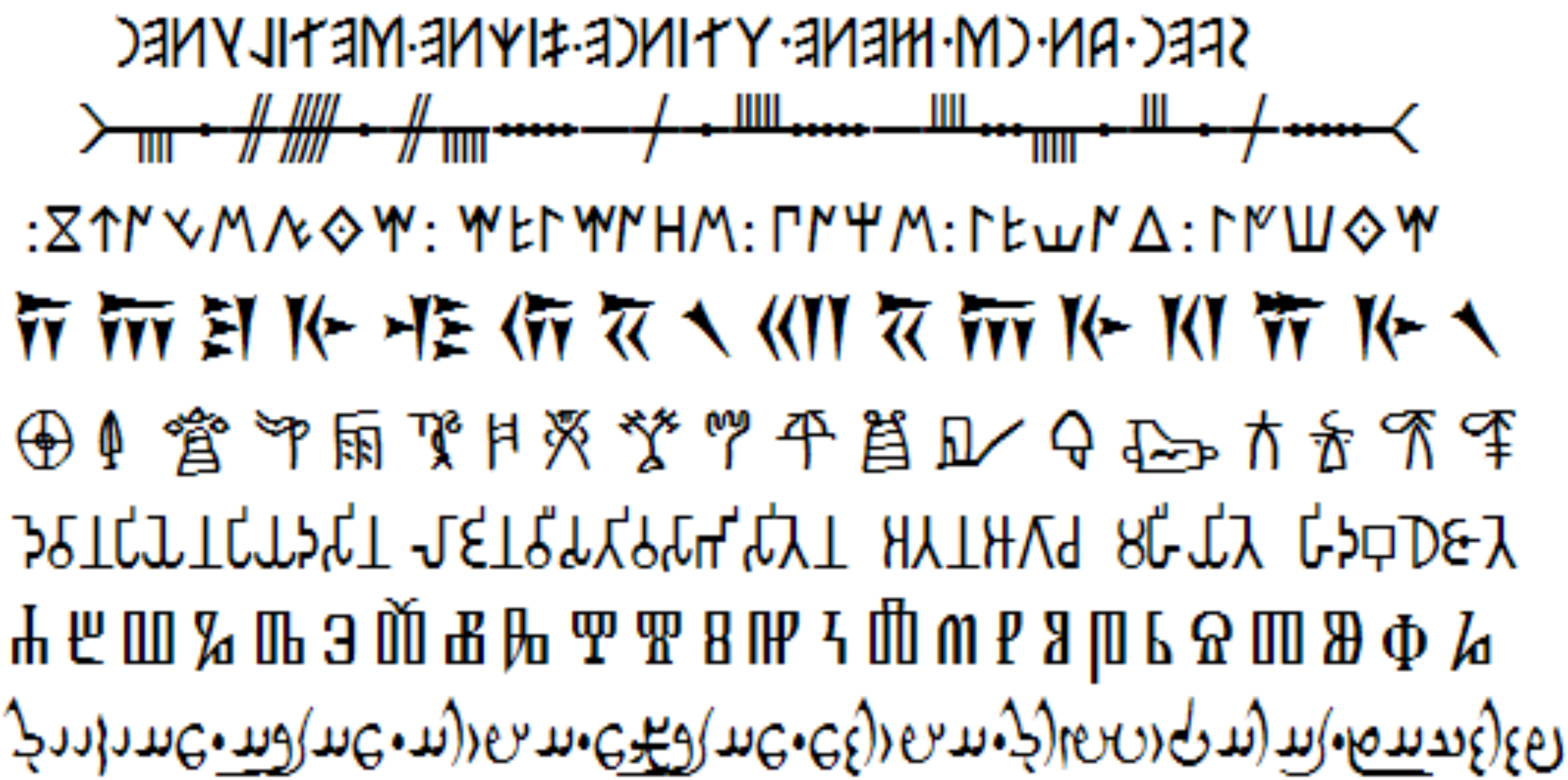
phonetician /ˌfəʊnəˈtɪʃən/ dog /dɒg/ bird /bɜːd/

И҆ речѐ бг҃ъ: да бꙋ́детъ свѣ́тъ. И҆ бы́сть свѣ́тъ.

א בְּרֵאשִׁית בָּרָא אֱלֹהִים אֵת הַשָּׁמַיִם וְאֵת הָאָרֶץ׃

𐌰 𐌱 𐌲 𐌳 𐌴 𐌵 𐌶 𐌷 𐌸 𐌹 𐌺 𐌻 𐌼 𐌽 𐌾 𐌿 𐍀 𐍁 𐍂 𐍃 𐍄 𐍅 𐍆 𐍇 𐍈 𐍉

अथ कालेन महता स मत्स्यः सुमहानभूत् ।




1 gigabyte of storage 20 years ago!

Some terminology

Some terminology

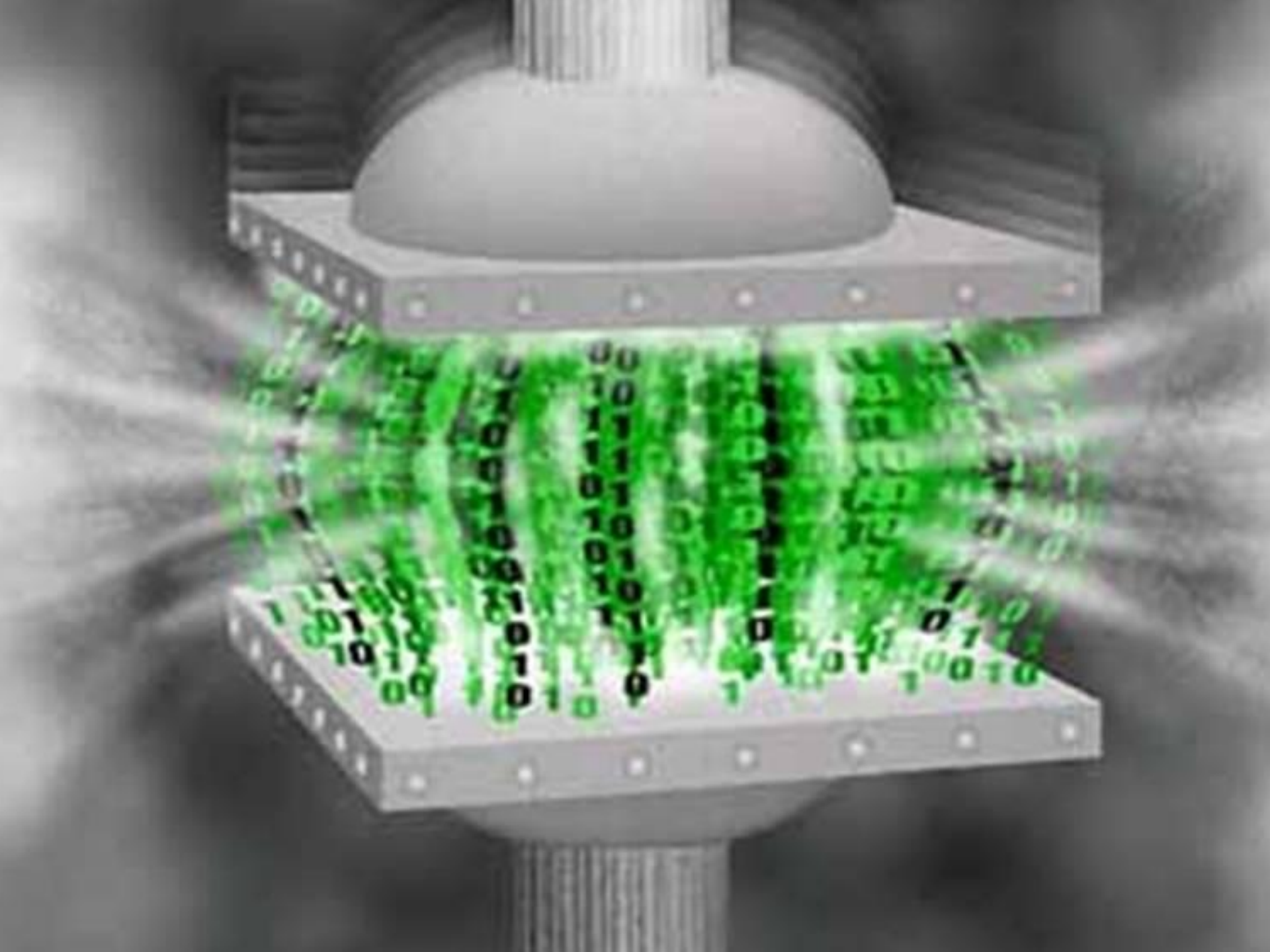
Up to this point we have been talking about data in either bits or bytes.

1 byte = 8 bits

While this is the correct way to talk about data, sometimes it is a bit inefficient.

Therefore, we use prefixes to given an order of magnitude. Much the same way we do with the metric system.

Prefix	Symbol	1000^m	10^n	Decimal
yotta	Y	1000^8	10^{24}	1 000 000 000 000 000 000 000 000
zetta	Z	1000^7	10^{21}	1 000 000 000 000 000 000 000
exa	E	1000^6	10^{18}	1 000 000 000 000 000 000
peta	P	1000^5	10^{15}	1 000 000 000 000 000
tera	T	1000^4	10^{12}	1 000 000 000 000
giga	G	1000^3	10^9	1 000 000 000
mega	M	1000^2	10^6	1 000 000
kilo	k	1000^1	10^3	1 000
hecto	h	$1000^{2/3}$	10^2	100
deca	da	$1000^{1/3}$	10^1	10
		1000^0	10^0	1



Data Compression

Why compress data?

Storage, transmission within PC/over network

Data Compression

What is data compression?

Reducing physical size of information blocks

Data Compression

Compression ratio

Tells us how much compression occurs. Number between 0 and 1

Lossless versus lossy compression

Images, sound files, videos

Database of names, numbers

compressed = ratio * uncompressed

ratio = compressed/uncompressed

Text Compression

Examine three types of text compression:

Keyword encoding

Run-length encoding

Huffman encoding

Keyword Encoding

Frequently used words replaced by a single character --> Reversible

Word	Symbol
as	^
the	~
and	+
that	$
must	&
well	%
these	#

The human body is composed of many independent systems, such as the circulatory system, the respiratory system, and the reproductive system. Not only must all systems work independently, but they must interact and cooperate as well. Overall health is a function of the well being of separate systems, as well as how these separate systems work in concert.

Keyword Encoding

Frequently used words replaced by a single character --> Reversible

Word	Symbol
as	^
the	~
and	+
that	$
must	&
well	%
these	#

The human body is composed of many independent systems, such **^** the circulatory system, **~** respiratory system, **+ ~** reproductive system. Not only **&** all systems work independently, but they **&** interact and cooperate **^ %**. Overall health is a function of **~ %** being of separate systems, **^% ^** how **#** separate systems work in concert.

Keyword Encoding

Frequently used words replaced by a single character --> Reversible

Word	Symbol
as	^
the	~
and	+
that	$
must	&
well	%
these	#

Reduced from 352 to 317
Compression ratio: 317/352 = 0.9

Is this efficient?

Keyword Encoding

Frequently used words replaced by a single character --> Reversible

Word	Symbol
as	^
the	~
and	+
that	$
must	&
well	%
these	#

Drawbacks:

Symbols used for encoding must not appear in the text

'The' & *'the'* needs to be represented by different symbols

Would not gain anything by encoding *'a'* and *'I'*

Most frequently used words are often short

Run-Length Encoding

Also known as *recurrence coding*

Encoding a single character that is repeated over and over again

For example: replacing ‘AAAAAAA’ with a ‘*’ : *A7

Drawbacks?

Uses: DNA sequences, simple images

Lossy or lossless compression?

Huffman Encoding

Variable bit lengths to represent characters:

a --> Binary 01100001 – 8 bits

Why would character *X* take up as many bits as *a*?

Represent it using *5 bits* instead

Saving space:

Frequently appearing characters are represented by shorter bit lengths

Huffman Encoding

Huffman Code	Character
00	A
01	E
100	L
110	O
111	R
1010	B
1011	D

DOORBELL

D= 1011 O= 110 O=110…

1011 110 110 111 101001100100

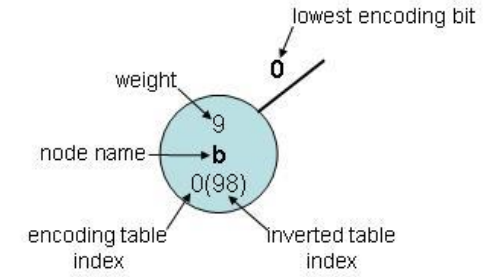
If we used fixed size bit string: 64 bits

With Huffman encoding: 25 bits

Compression ratio: 25/64 = 0.39

What about the decoding process?

Huffman Encoding Tree



Key to Huffman Encoding Tree Diagram

bcbbbbbbaacaabbcade