SEENG1050

Web Technologies

Lecture 10: Encoding, Compression

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

FACULTY OF ENGINEERING AND BUILT ENVIRONMENT

Lecture Overview

- Encoding
  - ASCII, Unicode
  - Error Checking: Parity, CRC
- Data Compression
  - Run-length, Huffman, LZW, gzip
  - Limits of data compression

Encoding

- In general: The conversion of information into another form
- For computers: The conversion of information into binary data
  - 1s and 0s = bits
- Encoding is not new:
  - The Telegraph and Morse Code...

Encoding - Morse Code

- 1836, Samuel Morse

- Alphabet:
  - Letters $\Rightarrow$ dots (short click) and dash (long clicks)
  - Spaces (between words) $\Rightarrow$ pauses

- Customers don’t need to know the rules!
  - Only the operator

http://en.wikipedia.org/wiki/Morse_code

Encoding - Morse Code

A	.-	G	--.	M	--	S	...	Y	-.--	4	....-
B	-...	H	....	N	-.	T	-	Z	--..	5	.....
C	-.-.	I	..	O	---	U	..-	0	-----	6	-....
D	-..	J	.---	P	.--.	V	...-	1	.----	7	--...
E	.	K	-.-	Q	--.-	W	.--	2	..---	8	---..
F	..-.	L	.-..	R	.-.	X	-..-	3	...--	9	----.

- Based on the distribution of letters in daily usage
- Short codes are assigned to more frequent letters
- Overall, fewer dots-dashes are needed to encode (and thus transfer) information

Encoding - Internet Encoding – ASCII

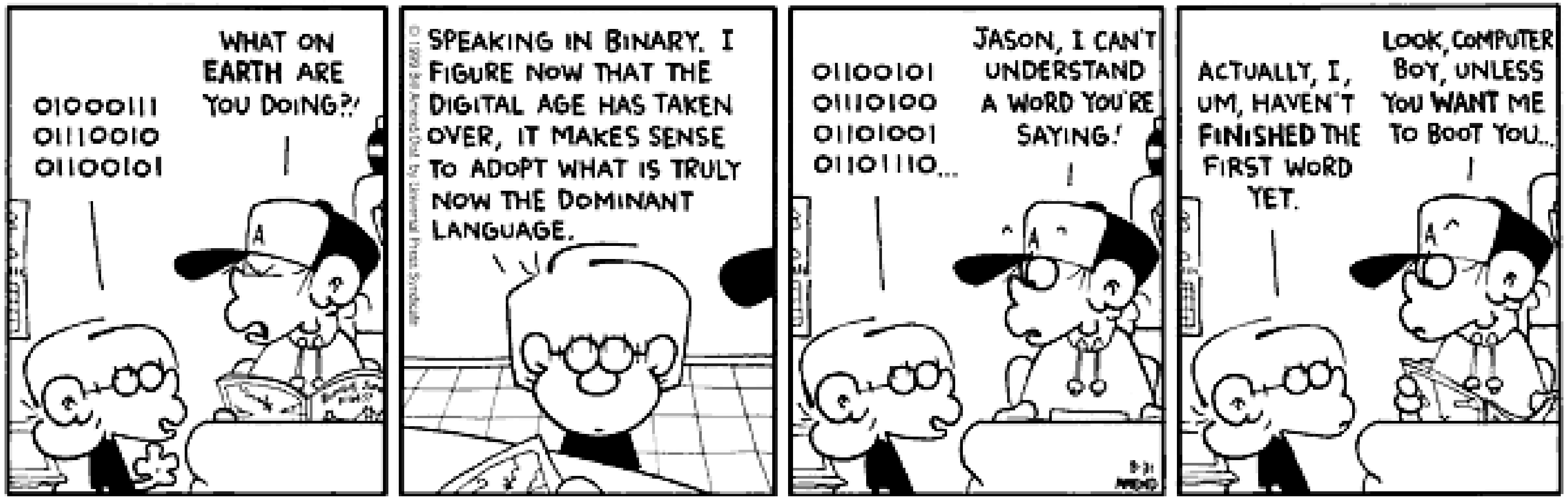
- Alphabet = {0, 1}
- Most common encoding(?) = **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
  - Encodes English characters – uppercase and lowercase letters, numbers, punctuation, space and some “special characters” (tab, return, linefeed, etc.)
  - Uses 7 bits / character $\Rightarrow$ 128 values
- Examples:

  ‘A’ = 1000001, ‘B’ = 1000010, ‘C’ = 1000011,

  ‘a’ = 1100001, ‘b’ = 1100010, ‘!’ = 0100001

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	□



What is Jason saying?

Encoding: Internet Encoding – Unicode

- ASCII only covers “English characters”
- Unicode supports over 1 million “characters” covering nearly all known languages
  - UTF-32 – uses 32 bits per character
  - UTF-16 – uses 16 bits per character
  - UTF-8 – uses 8 bits per character (compatible with ASCII)
  - http://www.unicode.org/versions/Unicode4.1.0/

Compression

- “Storing data in a format that requires less space than usual”
- Used to increase the amount of data that can fit on a storage device
  - Less important these days
- Used to decrease the number of bits needed to send information over a network connection
  - Still very important
  - Trades transmission time (expensive) for CPU time (cheap)

Compression - Morse Code

- Compression is not new: Morse code…

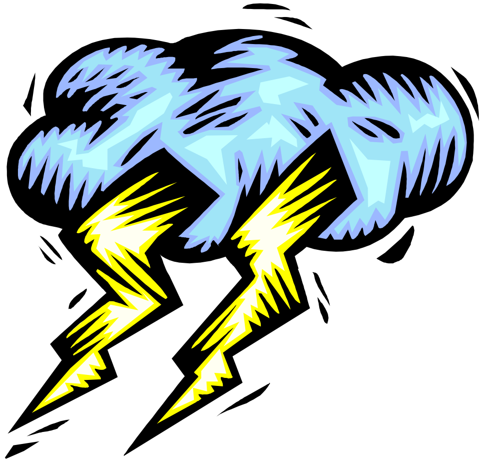
A .-	G --.	M --	S ...	Y -.--	4 ....-
B -...	H	N -.	T -	Z --..	5 .....
C -.-.	I ..	O ---	U ..-	0 -----	6 -....
D -..	J .---	P .--.	V ...-	1 .----	7 --...
E .	K -.-	Q --.-	W .--	2 ..---	8 ---..
F ..-.	L .-..	R .-.	X -..-	3 ...--	9 ----.

- Based on the distribution of letters in daily usage
- Short codes are assigned to more frequent letters
- Overall, fewer dots-dashes are needed to encode (and thus transfer) information

Compression and Archive

- A compressor compresses a single **file**
  - E.g., compress, gzip, bzip2, zip
- An archiver combines several input files into a single output file
  - Many archivers compress the individual files before combining
    - E.g., arc, arj, lha, zip, zoo, rar, jar, war
  - Some archivers do not
    - E.g., tar, ar
    - A compressor is often applied to the result – e.g., tgz
- Most archivers/compressors add error checking

Error Checking



- Things go wrong:
  - Faults in wiring
  - Interference from other wires
  - Interference from other devices
  - Interference from natural phenomena (lightning, sun spots)
- … can change the value of one or more bits

Error Checking - Parity

- Byte = bits needed to store a character = 8
  - But ASCII only uses 7 bits!
- 8th (parity) bit can be used for error checking
  - Doesn't ***only*** apply to ASCII
- Add all the 1s in the 7 bit string:
  - Even-parity: sum is odd $\Rightarrow$ 8^{th} bit = 1
    sum is even $\Rightarrow$ 8^{th} bit = 0
  - Odd-parity: sum is odd $\Rightarrow$ 8^{th} bit = 0
    sum is even $\Rightarrow$ 8^{th} bit = 1
- Even parity is mostly used.
- Used for keyboards, serial lines, etc.

Error Checking - Parity

- Example:
  1000011 = ‘C’ even-parity $\Rightarrow$ 11000011
  «interference»
  1000010 = ‘B’ 11000010 $\Rightarrow$ parity error
- Example:
  1000011 = ‘C’ even-parity $\Rightarrow$ 11000011
  «interference»
  0000000 = EOS 00000000 $\Rightarrow$ parity okay
- The Internet uses more powerful error checking…

Error Checking - Cyclic Redundancy Checking

Frame/Message: 11 01 01 10 11
Generator/Key : 10 01 1

Message after appending 0 CRC: **11 01 01 10 11 0000**

```
10 01 1| 11 01 01 10 11 0000
         10 01 1
         -------
          1 00 11
          1 00 11
          -------
            00 00 1
            00 00 0
            -------
              0 00 10
```

Transmitted message:

11 01 01 10 11 1110

Continue this process until you get the final remainder 1110

Computer Networks by Andrew S. Tanenbaum

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Error Checking - Cyclic Redundancy Checking

- The algorithm for computing the checksum:

- Let **r** be the degree of the Key/Generator G(x). Append **r** zero bits to the lower order end of the frame so it corresponds to x^r M(x).

- Divide the bit string corresponding to $x^r M(x)$ by the bit string corresponding to G(x) using modulo 2 division.

- The final remainder of the division is the checksum.

Computer Networks by Andrew S. Tanenbaum

Error Checking - Cyclic Redundancy Checking

- Knowing the message and the generator,
  - Receiver can recalculate the checksum and so be aware of errors.
  - A 16 bit generator (1100000000000001 is very good) will pick up double errors in a frame of 32,767 bits.
  - Note: G(x) = 11 is equivalent to odd-parity checksum.
  - If -1 is a root of G(x), then all errors consisting of an odd number of bits are detected.

Run-length Encoding compression

- Assumption: consecutive elements (bit, character, etc.) are likely to be the same
- Useful for encoding cartoons, or black and white images like faxes
- Repeating characters are replaced with a shorter representation

aaaaaaa
aabcdee
eeeeeee
efggghi
iijjjkk
kkkkkkk

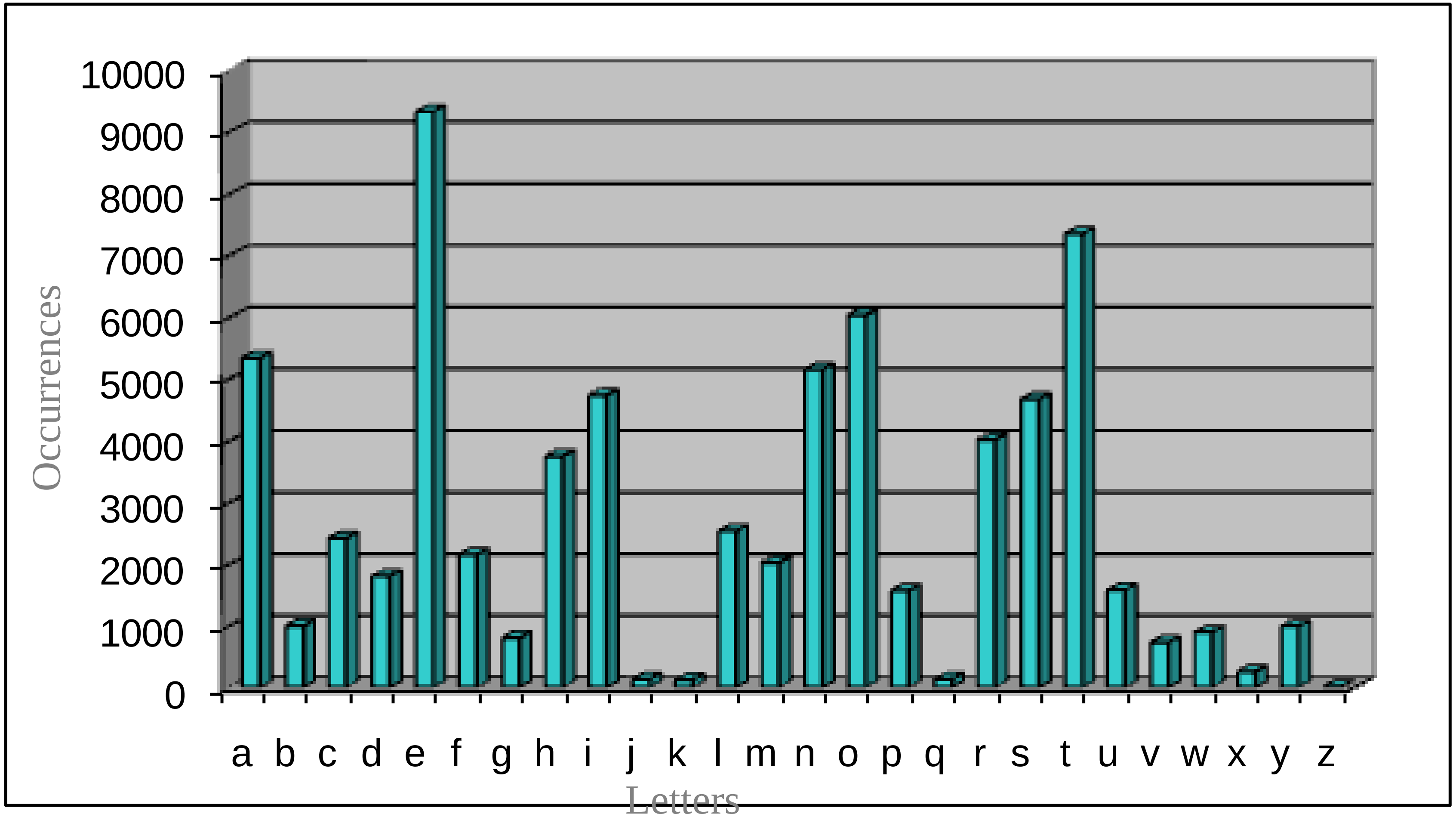
- Explicitly encode values

9a1b1c1d10e1f3g1h3i3j9k

22/42 bytes

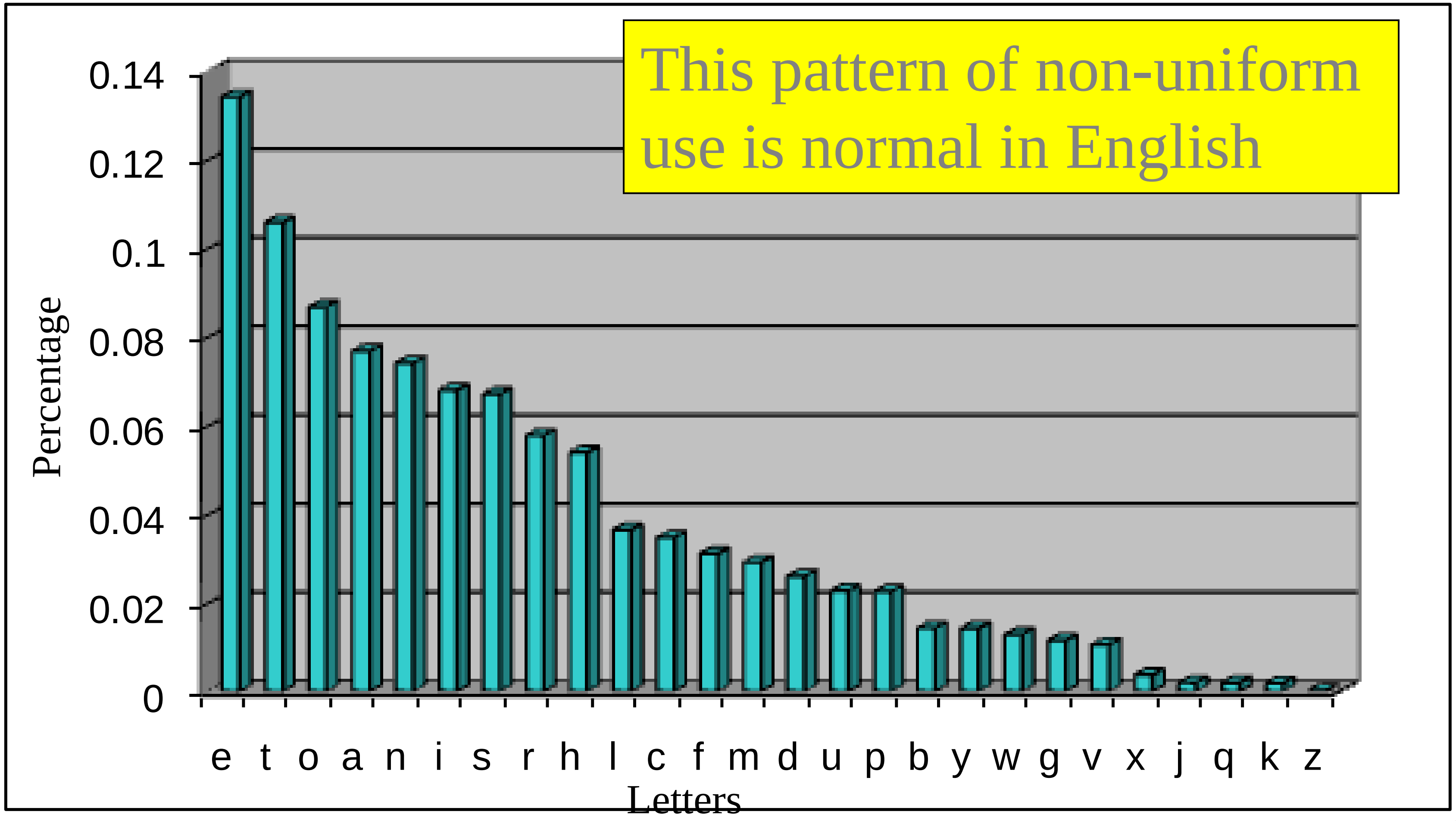
Statistical Compression

Aaron Quigley 1/5/2000: case ignored



Statistical Compression

Aaron Quigley 1/5/2000: case ignored



Huffman Encoding

- A statistical data compression technique
- Reduces the average code length used to represent the symbols of an alphabet
- To construct a Huffman code:
  1. Perform an analysis like that shown on the previous slides – rank all symbols by frequency of occurrence
  2. Repeatedly combine the two symbols of the **lowest probability** to form a new composite symbol – this builds **a binary tree** where each node is labelled with the frequency of all nodes beneath it
  3. Follow the “trail” to each leaf – left is `0` right is `1`
  4. This trail for each letter is it’s Huffman Coding

Huffman Encoding

Letter	Occurrence	Percentage
e	9338	0.408344
a	5345	0.233733
n	2420	0.105825
f	2157	0.094324
d	1808	0.079062
b	997	0.043598
g	803	0.035115
Total	22868	100%

First Combine
forms bg

Huffman Encoding

Letter	Occurrence	Percentage
e	9338	0.408344
a	5345	0.233733
n	2420	0.105825
f	2157	0.094324
d	1808	0.079062
(bg)	1800	0.078712
Total	22868	100%

Next Combine forms d and (bg)

Huffman Encoding

Letter	Occurrence	Percentage
e	9338	0.408344
a	5345	0.233733
n	2420	0.105825
f	2157	0.094324
d(bg)	3608	0.157775
Total	22868	100%

Next Combine
forms n and f

Huffman Encoding

Letter	Occurrence	Percentage
e	9338	0.408344
a	5345	0.233733
nf	4577	0.200148
d(bg)	3608	0.157775
Total	22868	100%

Next Combine
forms nf and d(bg)

Huffman Encoding

Letter	Occurrence	Percentage
e	9338	0.408344
a	5345	0.233733
((nf)(d(bg)))	8185	0.357923
Total	22868	100%

Next Combine
forms a and ((nf)(d(bg)))

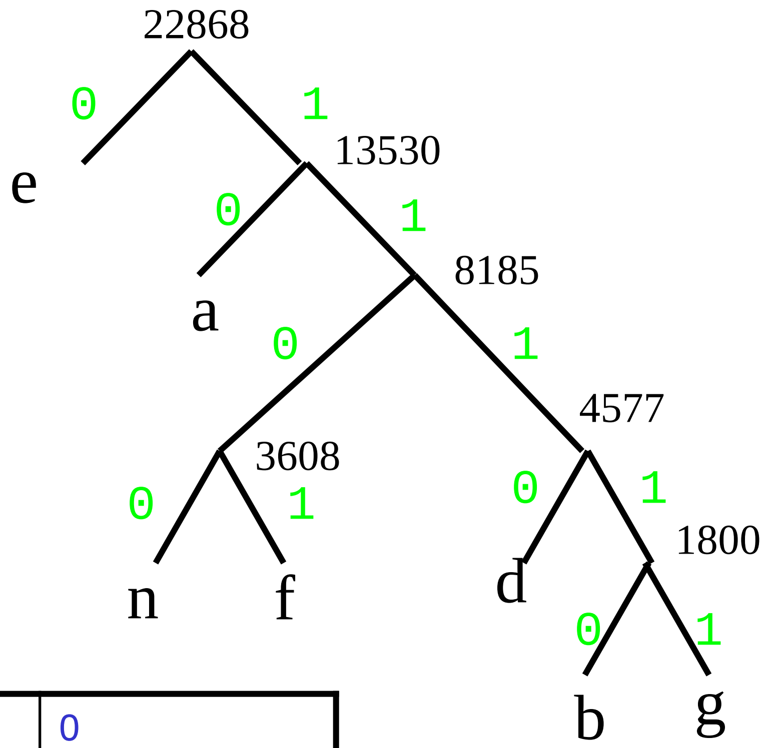
Huffman Encoding

Letter	Occurrence	Percentage
e	9338	0.408344
a((nf)(d(bg)))	13530	0.591656
Total	22868	100%

Finally Combine
forms e and a((nf)(d(bg)))

Huffman Encoding

Letter	Occurrence	Percentage
e	9338	0.408344
a	5345	0.233733
n	2420	0.105825
f	2157	0.094324
d	1808	0.079062
b	997	0.043598
g	803	0.035115
Total	22868	100%

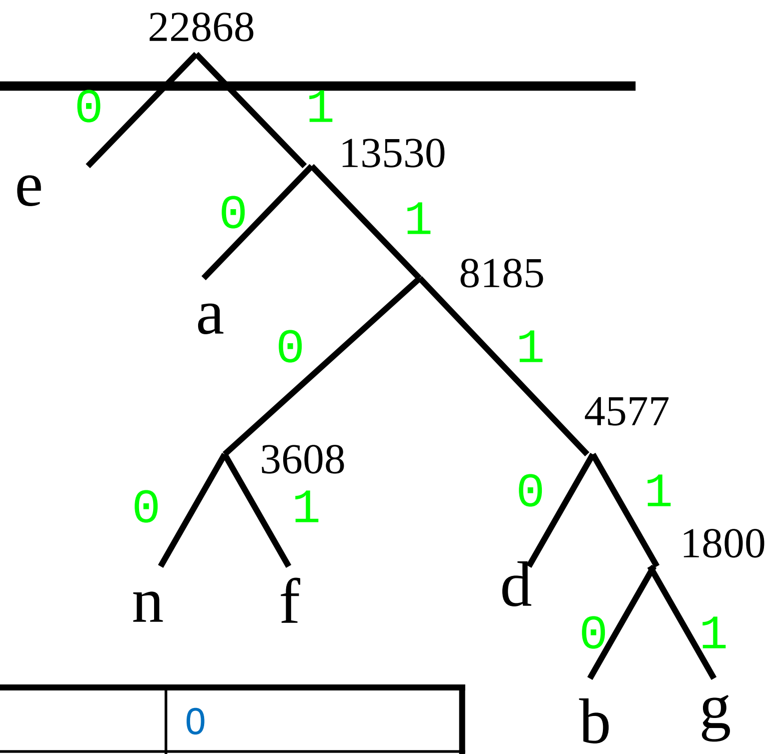


e(a((nf)(d(bg))))

e	0
a	10
n	1100
f	1101
d	1110
b	11110
g	11111

Huffman Encoding

Letter	Occurrence	Percentage
e	9338	0.408344
a	5345	0.233733
n	2420	0.105825
f	2157	0.094324
d	1808	0.079062
b	997	0.043598
g	803	0.035115
Total	22868	100%



e	0
a	10
n	1100
f	1101
d	1110
b	11110
g	11111

Try decoding this:

1111010110010110010

b

Huffman Encoding

- ASCII encoding uses 8 bits per character
- Here we use 19 bits for 6 characters ≈ 3.1667 bits per character
  - compression 1-19/48 ≈ 60%
- For a given frequency distribution, there are many possible Huffman codes, but the total compressed length will be the same
- http://www.compressconsult.com/huffman/

Adaptive Encoding

- Encoding adapts to properties of message
- Calculate letter frequencies for this message **at the start**
  - Easy to implement
  - Requires a decoding table to be included with data
  - Good for long data string

Non-adaptive Encoding

- For encoding a fixed table is used
- Expected frequency of letters are used
- Does not need a decoding table to be sent
- Good for short data string

A third approach can be used.

A Third Approach of Encoding

- Gradually learn from the data string and adapt a table.
- Start as a non-adaptive encoding with a fixed table.
- Keep updating the table.
- For decompression the same process is repeated.
- Decoding table is not required to be sent.

Substitutional Compressors

- Based on the assumption that a phrase within a message is likely to occur several times
- Phrases are entered into a dictionary
- Occurrences of a phrase use the dictionary index, rather than the phrase itself
- Dictionary can be seeded with common phrases
- Dictionary can be expanded as the message is encoded to include new phrases
- Dictionary can be implicit

Substitutional Compressors – LZW

- Lempel-Ziv-Welch
- The compression algorithm is used by GIF
  - Patented by Unisys
- Initial dictionary = all single characters
- For each phrase that matches dictionary
  - Write index of phrase
  - Add phrase + next character to dictionary

- http://marknelson.us/1989/10/01/lzw-data-compression/

LZW Compression Algorithm

- Source: http://marknelson.us/1989/10/01/lzw-data-compression/

```
STRING = get input character
WHILE there are still input characters DO
    CHARACTER = get input character
    IF STRING+CHARACTER is in the string table then
        STRING = STRING+character
    ELSE
        output the code for STRING
        add STRING+CHARACTER to the string table
        STRING = CHARACTER
    END of IF
END of WHILE
output the code for STRING
```

LZW Compression Algorithm

Input String = /WED/WE/WEE/WEB/WET – 19bytes

String	Output Code	New code value	New String
/W	/	256	/W
E	W	257	WE
D	E	258	ED
/	D	259	D/
WE	256	260	/WE
/	E	261	E/
WEE	260	262	/WEE
/W	261	263	E/W
EB	257	264	WEB
/	B	265	B/
WET	260	266	/WET
EOF	T		

Output String = /WED256E260261257B260T – 12 bytes

LZW Decompression Algorithm

Original String = /WED/WE/WEE/WEB/WET

First Unknown Pair/Set

Compressed String = /WED256E260261257B260T

LZW Decompression Algorithm

Original String = /WED/WE/WEE/WEB/WET

Compressed String = /WED256E260261257B260T

First Unknown Pair/Set

Decompression = /WED/WE → Step 1

Decompression = /WED/WE/WE → Step 2

Decompression = /WED/WE/WEE/ → Step 3

Decompression=/WED/WE/WEE/WEB/WET → Final Step

Substitutional Compressors – gzip

- A variation on the LZ77 algorithm
  - Not patented by anyone
- Dictionary = a window of the most recently processed part of the message
- If a phrase appears in the window, then output its **(negative) offset and length**
- If a phrase does not appear, then output its characters
- Apply Adaptive Huffman Encoding to output!
- http://www.gzip.org/algorithm.txt

Substitutional Compressors – gzip

- Example: Compress ABABCBABAB

Characters	Output	Window
A	A	A
B	B	AB
AB	(-2,2)	ABAB
C	C	ABABC
BAB	(-4,3)	ABABCBAB
AB	(-2,2)	ABABCBABAB

- Note: blahblahblahblah $\Rightarrow$ blah(-4,12) !!

Substitutional Compressors – gzip

- Original String: ABABCBABAB

  - Compressed String: AB(-2,2)C(-4,3)(-2,2)
  - Decompressed String: AB AB C BAB AB

The Best Data Compressor?

- Many factors
  - Speed of method
  - Computing resources required
  - Compression ratio

Compression Techniques

- Lossless Compression Techniques
  - In lossless data compression, the integrity of the data is preserved. The original data and the data after compression and decompression are exactly **the same** because, in these methods, the compression and decompression algorithms are exact inverses of each other: no part of the data is lost in the process. Redundant data is removed in compression and added during decompression. Lossless compression methods are normally used when we cannot afford to lose any data.
- Lossy Compression Techniques

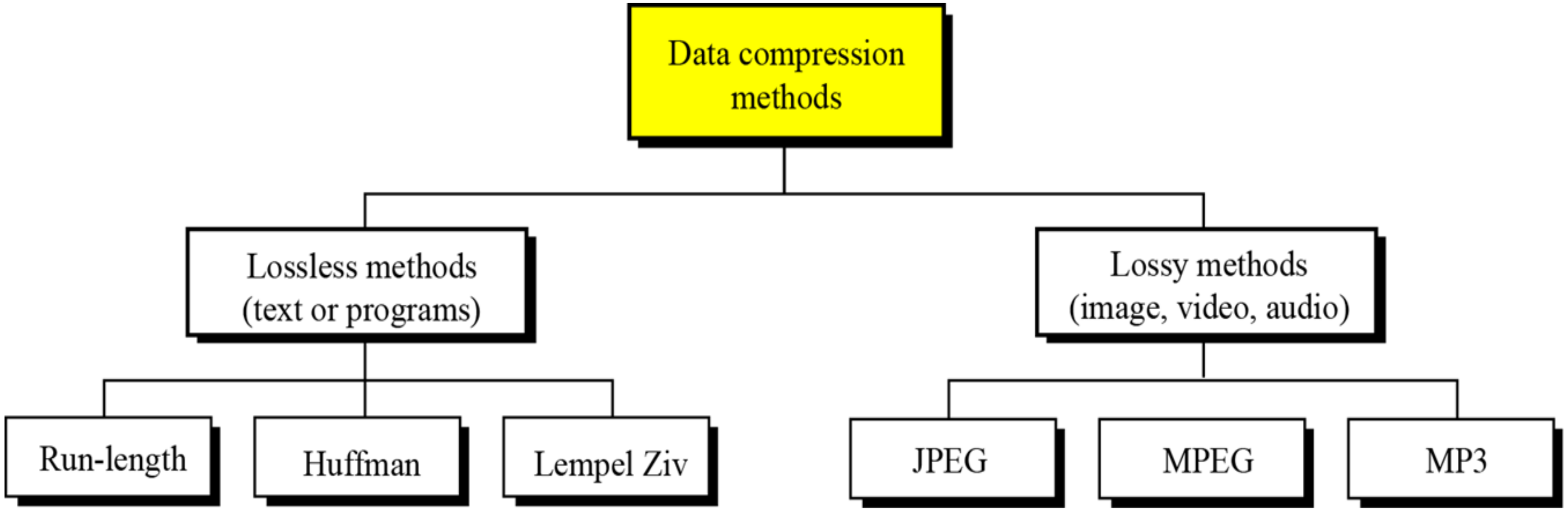
Compression Techniques

- Lossless Compression Techniques
  - In lossless data compression, the integrity of the data is preserved. The original data and the data after compression and decompression are exactly **the same** because, in these methods, the compression and decompression algorithms are exact inverses of each other: no part of the data is lost in the process. Redundant data is removed in compression and added during decompression. Lossless compression methods are normally used when we cannot afford to lose any data.
- Lossy Compression Techniques
  - Our eyes and ears cannot distinguish subtle changes. In such cases, we can use a **lossy** data compression method. These methods are cheaper—they take less time and space when it comes to sending millions of bits per second for images and video.

Compression Techniques

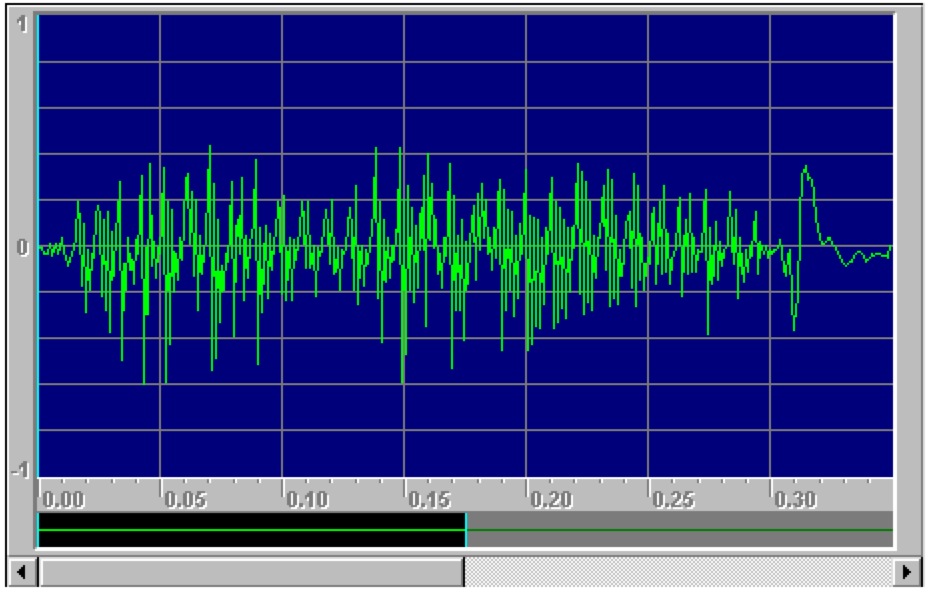
- Lossless Compression Techniques
  - Data (D) → Compression → Compressed Data (C) → Decompression → Data D
    - `uncompress(compress(`*data*`)) ==` *data*
  - Statistical data
  - Text documents
  - Executable file
- Lossy Compression Techniques
  - Data (D) → Compression → Compressed Data (C) → Decompression → Decompressed Data (E)
    - `uncompress(compress(`*data*`))` ≈ *data*
  - Audio
  - Video

Compression Techniques



Digital Audio

- Sound is *analogue*
  - potentially infinite number of levels for sound signal



http://electronics.howstuffworks.com/analog-digital2.htm

Digital Audio

- Want to convert into *digital* format
  - finite number of levels
  - some information is lost in conversion
- Analogue-to-Digital Converter (ADC)
  - converts analogue signal (sound wave) to produce a digital version
    - Sampling – periodic snapshots
    - Quantisation – number of levels
  - CD stereo– 44.1kHz (44,100 time per second) sampling rate,16 bits amplitude (2^{16} or 65,536 division of amplitude) -

    = 1400 Kb/second (transfer rate) [storage: 10.1 Mb per minute]

    -- AM radio (mono) – 22.05 KHz – 8 bit -22 Kb/sec [1.26 Mb]

    -- Telephone (mono) – 11.025KHz – 8 bit – 11 kb/sec [630 Kb]

(60min * 60) secs x 44100 samp/sec x 2 bytes/samp x 2 channels = 635,040,000
About 600 Mbytes (typical CD)

MP3

- Short for MPEG 1 Layer III
  - audio component of MPEG's movie compression standard
  - Sampling: 44.1 kHz is common
  - Bit rate: 128 kbits/s (1/11 of CD) at a compression ratio of 11:1 or 320 kbits/s
- Compression is based on *perceptual audio encoding* or *perceptual noise shaping*
  - Human hearing range (2-5kHz is most sensitive)
  - Psychoacoustic behaviour (loud masks soft)
    - Psychoacoustic model (PAM)
  - Stereo Effects
  - Huffman encoding is applied at the end
    - Nor for compression, for encoding

MP3

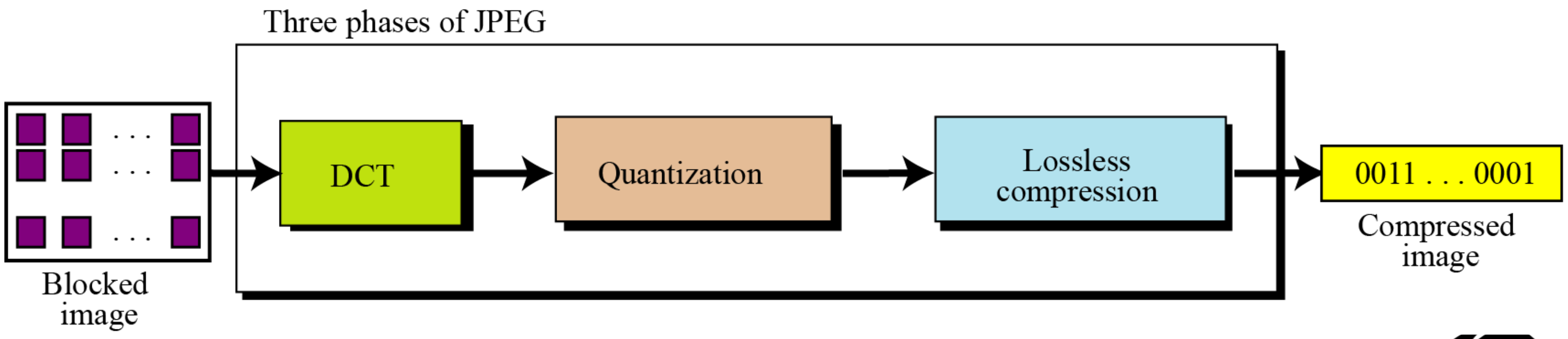
- Short for MPEG 1 Layer III
  - audio component of MPEG’s movie compression standard
  - Sampling: 44.1 kHz is common
  - Bit rate: 128 kbits/s (1/11 of CD) at a compression ratio of 11:1 or 320 kbits/s
- Compression is based on *perceptual audio encoding* or *perceptual noise shaping*
  - Human hearing range (2-5kHz is most sensitive)
  - Psychoacoustic behaviour (loud masks soft)
    - Psychoacoustic model (PAM)
  - Stereo Effects
  - Huffman encoding is applied at the end
    - Nor for compression, for encoding

JPEG

- Joint Photographics Experts Group – committee that wrote the standard
- Relies on the human eye’s inability to discern small changes in colour
- Compression discards this redundant information
- Two parts:
  - Baseline lossy algorithm
  - Lossless algorithm is applied to this output

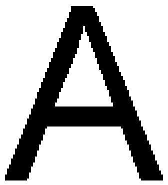
JPEG

- Steps in algorithm – *changes the picture into a linear set of numbers*
  1. Divides a picture into blocks of 8 x 8 pixel blocks
  2. Group pixels and perform discrete cosine transform
  3. Quantisation step – **loses** some info
  4. Encode with **lossless** Huffman technique
  5. Add file headers, decoding table and other decoding information to produce final file.

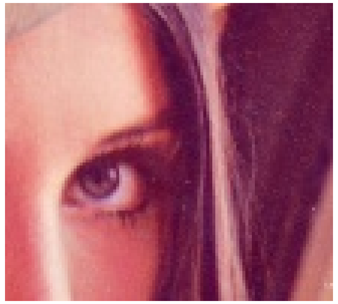


JPEG

- Limitations of JPEG compression
  - Doesn’t perform well with high contrast areas

Original  Quality: 0 

  - Multiple applications have an accumulative effect

Original  Max compression 

http://www.cs.cmu.edu/~chuck/lennapg/lenna.shtml

MPEG

- Moving Picture Experts Group
  - family of standards used for coding audio-visual information in a digital compressed format
  - There are a number of *phases* to the work of the MPEG organization. These are indicated by a number following “MPEG”.

    MPEG-1 - Video CD, MP3

    MPEG-2 - Digital Television, set top boxes, DVD

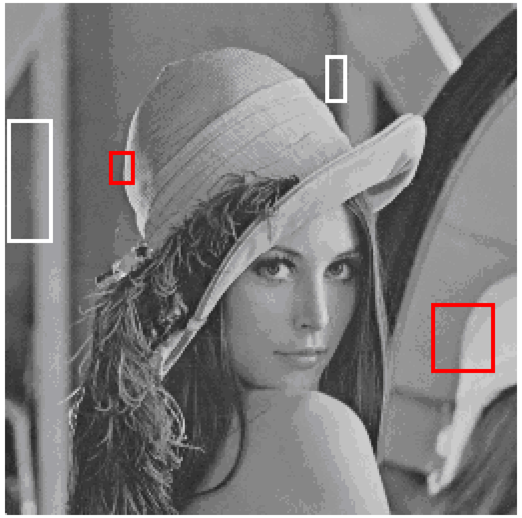
    MPEG-4 - multimedia for the fixed and mobile web

    MPEG-7 - description and search of audio and visual content

Fractal Compression

- Fractals are mathematical geometric models which, unlike other geometric models, have infinite detail contained within them
- The principle behind fractal based image compression is to find areas in the picture which are similar to other areas in the picture.

Fractal Compression



Two repeated areas

- Very slow compression, fast decompression
- Extremely high potential compression rate
- Technology is patented

Questions?