COMP9319 Web Data Compression and Search

Course Overview,

State-of-the-art Compressions

COMP9319 – The foundations of

- how different compression tools work.
- how to manage a large amount of data on small devices.

- how to search gigabytes, terabytes or petabytes of data.
- how to perform full text search efficiently without added indexing.
- how to query distributed data repositories efficiently (optional).

Course Aims

As the amount of Web data increases, it is becoming vital to not only be able to search and retrieve this information quickly, but also to store it in a compact manner. This is especially important for mobile devices which are becoming increasingly popular. Without loss of generality, within this course, we assume Web data (excluding media content) will be in XML and its like (e.g., HTML, JSON).

This course aims to introduce the concepts, theories, and algorithmic issues important to Web data compression and search. The course will also introduce the most recent development in various areas of Web data optimization topics, common practice, and its applications.

Course info

- Homepage: www.cse.unsw.edu.au/~cs9319
- Lectures: Tue 19:00-21:00, Wed 19:00-21:00
  - Weeks 1-5, 7-10 (no lectures in wk6)

Lecturer in charge

Raymond Wong

Office: K17 Level 2 (Room 213)

Email: wong@cse.unsw.edu.au

Ext: 55932

www.cse.unsw.edu.au/~wong

Lectures

- Supposed to be 2hr + 2hr f2f per week

- For 2020T2 week 1:
  - Live lecture (Collaborate) on Tue (today): 2hrs
  - Lecture recordings on Wed: 2hrs

- For 2020T2 **week 2 onwards**:
  - Lecture recordings 3hrs p/w
  - Live lecture (Collaborate) **1hr Wed 7-8pm** AEST

- *Live lectures will be recorded as well*

Lecture recordings

- Go through the scheduled topics in details
- Less problematic due to bad connections (from your side or my side)
- Less interruption due to Q&A
- **Note: we assume that you will watch the recordings every week; and attend & ask any questions at the live lectures / consultations**

Live lectures

- Lecture recordings are good but no interactions or Q&A
- Hence there is a one-hour live lecture to go through more examples, and/or answer any Q&A for the topics covered during the wk.
- **If you do not have problems with the lecture recordings, you do not need to attend the live lectures and instead, watch their recordings as well.**

Exercises

Exercises will be provided on WebCMS regularly.

- Brief solutions will be released one week later
- If you’re stuck, feel free to talk to us in the consultations.

No tutorials

- Have consultation slots instead
- Lecture Q&A will take place during the Collaborate live lectures
- Specific lecture / exercise / assignment questions can be addressed better in consultations
- Don’t leave your questions till very late, we won’t be able to address questions that stacked for many weeks within an hr

Consultation hours

- Week 2 - Week 10 (except wk 6)
  - **Mon** 11am-12pm AEST
  - **Thu** 1pm-2pm AEST
  - **Fri** 7pm-8pm AEST
- More consultation slots will be added during the week of assignment due
- Discuss any questions on exercises & assignments
- Answer questions on lecture materials

WebCMS Forum

- For short questions (such as clarification of assignment spec or lecture materials), you can post at the WebCMS Forum.
- Your peers, tutors, or myself can help answer your questions.
- For longer / complicated questions, better see us at the online consultations.

Readings

- No text book
- Slides will be provided / linked from the course homepage
- Core readings (papers) are also provided
- References / supplementary reading list can be found from the course homepage

Assumed knowledge

Official prerequisite of this course is COMP2521 / COMP1927 / COMP9024.

At the start of this course students should be able to:

- understand bit and byte operations in C/C++.
- write C/C++ code to read from/write to files or memory.
- produce **correct** programs in C/C++, i.e., compilation, running, testing, debugging, etc.
- produce readable code with clear documentation.
- appreciate use of abstraction in computing.

Learning outcomes

- have a good understanding of the fundamentals of text compression
- be introduced to advanced data compression techniques such as those based on Burrows Wheeler Transform
- have programming experience in Web data compression and optimization
- have a deep understanding of XML and selected XML processing and optimization techniques
- understand the advantages and disadvantages of data compression for Web search
- have a basic understanding of XML distributed query processing
- appreciate the past, present and future of data compression and Web data optimization

Assessment

```
a1          = mark for assignment 1      (out of 15)
a2          = mark for assignment 2      (out of 35)
asgts       = a1 + a2                    (out of 50)
exam        = mark for final exam        (out of 50)
okEach      = asgts > 20 && exam > 20    (after scaling)
mark        = a1 + a2 + exam
grade       = HD|DN|CR|PS  if mark >= 50 && okEach
            = FL           if mark <  50 && okEach
            = UF           if !okEach
```

One final exam

- One final exam (50 pts)
- If you are ill on the day of the exam, do not attend the exam – c.f. fit-to-site policy. Apply for special consideration asap.
- It's a 3-hr online exam, and you can complete it within 24hrs (covers all time zones).

Two assignments

- 1 small prog assignment (15 pts)
- 1 larger prog assignment (35 pts)
- Penalty for late assigt submission: 1% per hour (no acceptance after 3 days late).
- Advanced project in lieu of the assignments is possible. Pre-arrangement with the lecturer before end of week3 if interested.

Programming assignments

- First assignment is warm-up, relatively easier
- The 2nd assignment is larger in scale, and relatively challenging
- In addition to correctness, reasonable runtime performance is required
- All submitted code will be checked for plagiarism.

Tentative assignment schedule

#	Description	Due	Marks
1	Programming assignment 1 (fundamental)	Week 5	15%
2	Programming assignment 2 (compression and search)	Week 9	35%

In the past, students failed assigts because:

- *Plagiarism*
- Code failed to compile due to various reasons
- Program worked on windows but not CSE linux
- Late submission
- Program did not follow the spec
- Program failed auto-marking

Please do not enrol if you

- Not comfortable with COMP2521 / COMP1927 / COMP9024
- Cannot produce correct C/C++ program on your own
- Have poor time management
- Are too busy to attend lectures and catch up as soon as possible, if needed
- *Otherwise, from past experience, you are unlikely to perform well in this subject*

Tentative course schedule

Week	Lectures	Assignments
1	Introduction, basic information theory, basic compression	
2	More basic compression algorithms	
3	Adaptive Huffman; Overview of BWT	a1 released
4	Pattern matching and regular expression	
5	FM index, backward search, compressed BWT	a1 due; a2 released
6	-	
7	Suffix tree, suffix array, the linear time algorithm	
8	XML overview; XML compression	
9	Graph compression; Distributed Web query processing	a2 due
10	Optional advanced topics; Course Revision	

Summarised schedule

1. Compression
2. Search
3. Compression + Search
4. "Compression + Search" on Web text data
5. Selected advanced topics (if time allows)

distributed queries in database

QUESTIONS?

Questions to discuss (www)

- ① What (is data compression)
- ② Why (data compression)
- ③ Where

Compression

- Minimize amount of information to be stored / transmitted

- Transform a sequence of characters into a new bit sequence

无损压缩

  - same information content (for lossless)
  - as short as possible

Familiar tools

- Tools for
  - .Z
  - .zip
  - .gz
  - .bz2
  - …

A glimpse

raaabbcccdabbbbeee$

$\Downarrow$

ra3b2c4dab4e3

Run-length coding

- Run-length coding (encoding) is a very widely used and simple compression technique
  - does not assume a memoryless source
  - replace runs of symbols (possibly of length one) with pairs of (symbol, run-length)

to a pair (character, number)

RLE

map a series of characters

raaabbcccdaaaaabbbbeeeeeed$

ra3bbc4da5b4e6d$

Example: BWT

rabcabcababaabacabcabcabcababaa$

Example: BWT

rabcabcababaabacabcabcabcababaa$

aabbbbccacccrcbaaaaaaaaaaabbbbba$

Example: BWT+RLE

rabcabcababaabacabcabcabcababaa$

aabbbbccacccrcbaaaaaaaaaabbbbba$

⇓ reduce less or equals to 2.

aab4ccac3rcba10b5a$

HTTP compression

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
```

Cf. http://www.http-compression.com/

telerik

Developer Productivity | Team Productivity | Automated Testing | Web Content Management | About Telerik | Purchase | Free Trials

Log In or Register (Why?)

Search Site

Web UI Components & Controls
ASP.NET AJAX • MVC • Silverlight
Desktop UI Components & Controls
Windows Forms • WPF
Mobile UI Controls & Components
Windows Phone
Reporting & ORM
Telerik Reporting • OpenAccess ORM
Code Analysis & Refactoring
JustCode
Automated Testing & Mocking
WebUI Test Studio • JustMock
Support
Documentation, KBs, Tickets
Community
Forums, Blogs, Events
Telerik TV
Videos, Webinars, Interviews
Services
Training, Consulting

RadCompression for ASP.NET AJAX

Home > Developer Productivity > Products > ASP.NET AJAX Controls > Compression

Overview

RadCompression is a proud member of Telerik's AJAX family of performance optimization helper controls. It automatically ZIP-s the AJAX and WebService responses for even faster data transfers and improved page performance. The compression process is completely transparent to your client-side code (JavaScript or Silverlight) and your server-side code.

See Demos

This and 70+ other controls are part of
RadControls for ASP.NET AJAX

$999 Buy Now or Download Free Trial

Get this product + 7 more as part of
Premium Collection Bundle

- All Telerik UI for web/desktop
- Code Analysis and Refactoring tools
- Data Access Tools & Reporting Engine

$1299 Buy Now or Download Free Trial

Features

- Types of Compressed Content
- ViewState Compression
- AJAX Responses Performance Tests
- ViewState Compression Performance Tests

Add-ons for RadControls

- Visual Studio Plug-in for Automated Testing
- Visual Style Builder
- Visual Studio Extensions

Types of Compressed Content

Telerik ASP.NET AJAX Compression is not designed to be a complete replacement for other HTTP compression tools, such as the built-in HTTP Compression in IIS 7. Instead, it is designed to work with those existing tools to cover scenarios they usually miss - namely the compression of bits moving back and forth in AJAX (and now Silverlight) applications. Telerik ASP.NET AJAX

From: http://www.telerik.com/products/aspnet-ajax/compression.aspx

Choose Opera
The Web is changing

Compress data, Save money

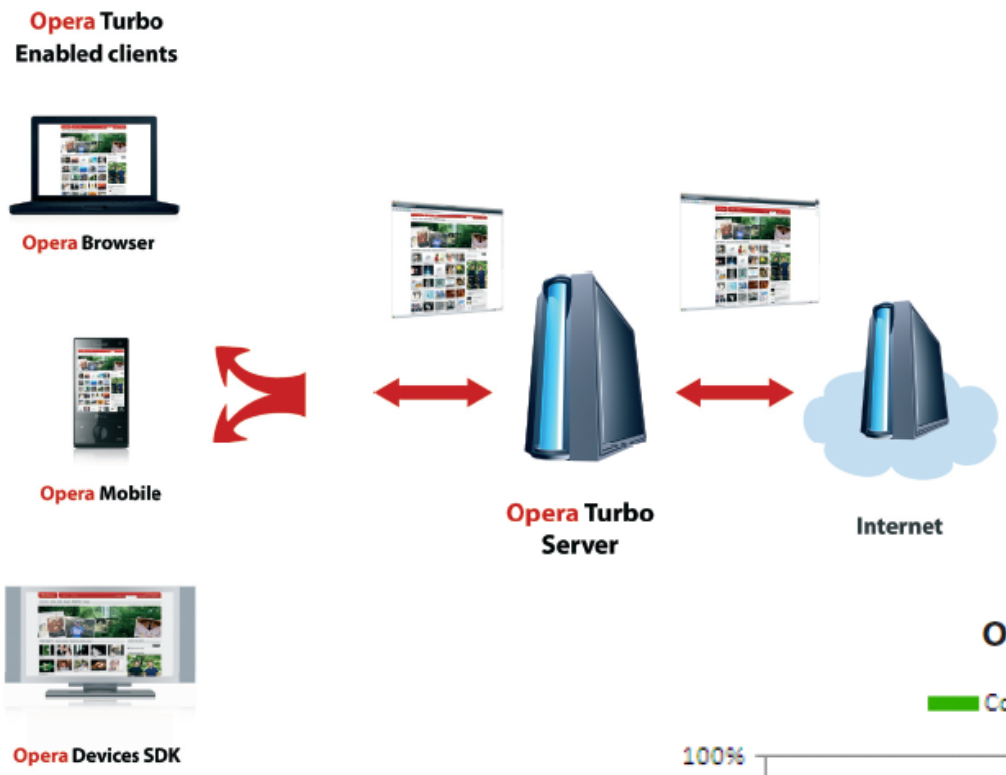
By Espen André Øverdahl. Wednesday, 27. October 2010, 08:18

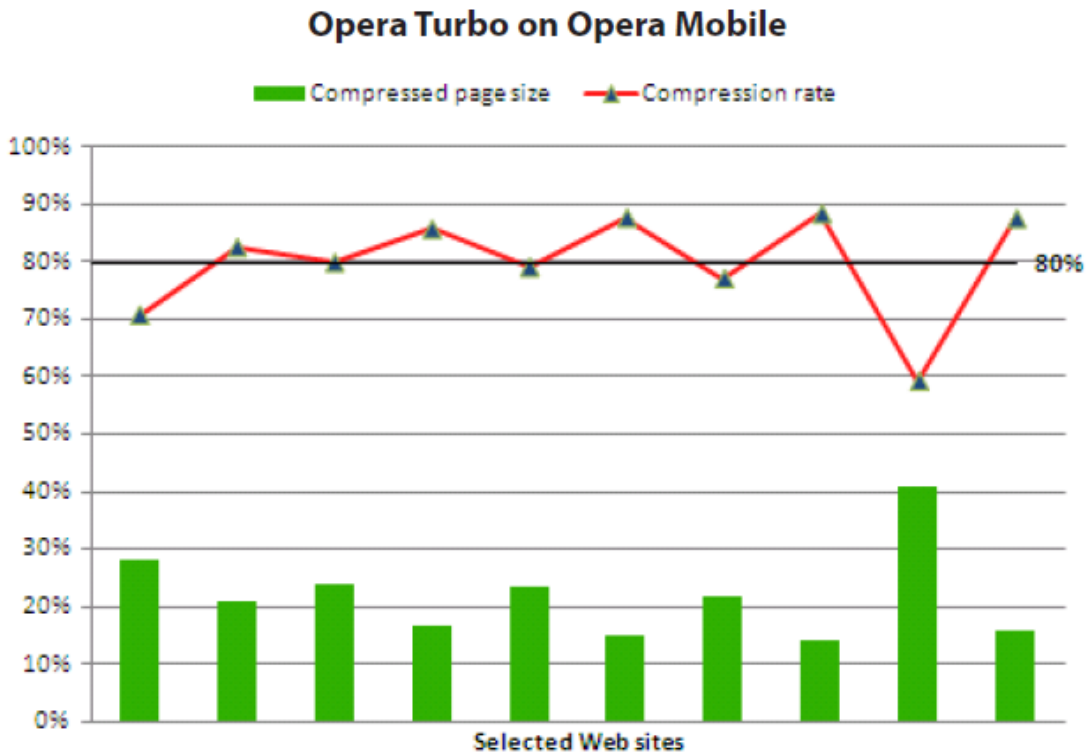
opera mini, opera turbo, smw, odd and even

In the latest State of The Mobile Web Report (PDF) issued today, you can read that Opera Mini saves consumers worldwide more than 2.2 billion USD each month on their mobile data bills. How? Opera Mini uses a unique compression technology, which reduces the size of web pages up to 90 percent before they are loaded on your mobile phone:

It's particular consumers in the United States and Nigeria who who benefit the most from Opera Mini's unique compression technology. If you are using your default phone browser, you might be thinking of making the switch. Not only will it save you money, but you will also load web pages faster and access your bookmarks and Speed Dial using Opera Link.

From: http://my.opera.com/chooseopera/blog/2010/10/27/compress-data-save-money





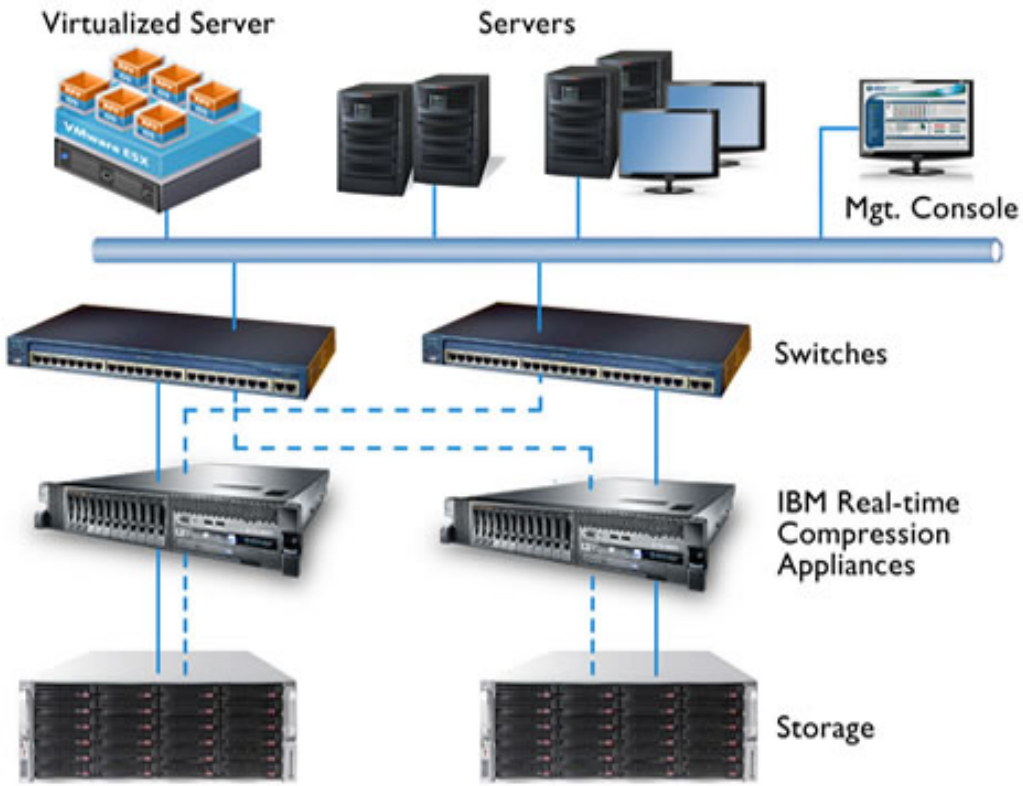
From: http://media.opera.com/media/b2b/Opera_Turbo_white_paper.pdf

Storwize

benefits

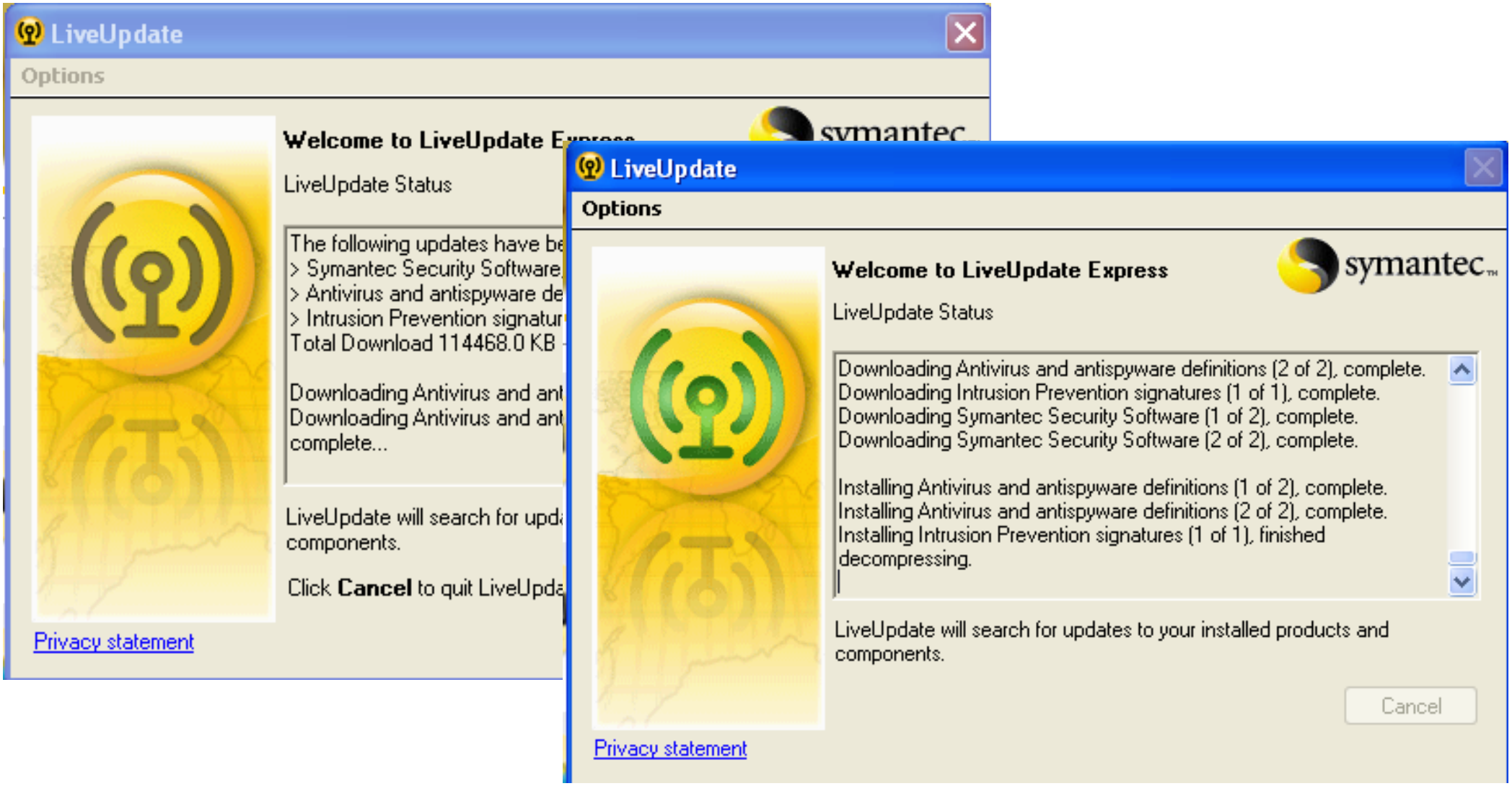
- Better Storage Utilization
- Reduces existing storage utilization up to 80%
- No performance degradation
- Lowers Capital and Operational Costs
- Better Energy Efficiency
- Less to store, power and cool

…



Source: from storwize.com

Anti-virus definitions & updates



Others

- Software updates

  e.g., Reg files, UI schemas / definitions
- Software configuration/database updates

  e.g., Virus database for anti-virus software
- Data streams

  e.g., RSS

Compression & patents

- e.g., STAC vs Microsoft

Microsoft Loses Case On Patent

By LAWRENCE M. FISHER and

A Federal court jury found the Microsoft Corporation guilty of patent infringement today and awarded $120 million in damages to a small California company that had accused Microsoft of appropriating its technology for increasing the storage capacity of

- e.g., United States Patent 5,533,051:

  the direct bit encode method of the present invention is effective for reducing an input string by one bit regardless of the bit pattern of the input string.

```
wong:Desktop wong$ ls -l image.jpg
-rwx------@ 1 wong  staff  671172 11 Feb 17:32 image.jpg
wong:Desktop wong$ gzip image.jpg
wong:Desktop wong$ ls -l image.jpg.gz
-rwx------@ 1 wong  staff  424840 11 Feb 17:32 image.jpg.gz
wong:Desktop wong$ mv image.jpg.gz image.jz
wong:Desktop wong$ gzip image.jz
wong:Desktop wong$ ls -l image.jz.gz
-rwx------@ 1 wong  staff  424932 11 Feb 17:32 image.jz.gz
wong:Desktop wong$ mv image.jz.gz image.jzz
wong:Desktop wong$ gzip image.jzz
wong:Desktop wong$ ls -l image.jzz.gz
-rwx------@ 1 wong  staff  425018 11 Feb 17:32 image.jzz.gz
wong:Desktop wong$
```

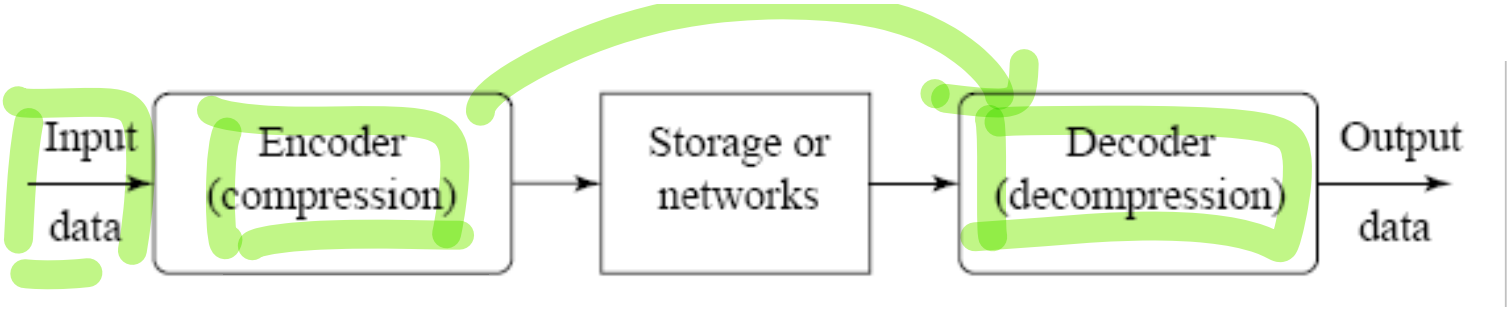
Similarity measure

If two objects compress better together than separately, it means they share common patterns and are similar.

From: Li, M. et al., “The similarity metric”, IEEE Transactions on Information Theory, 50(12), 2004

Overview

- Compression refers to a process of coding that will effectively reduce the total number of bits needed to represent certain information.



- Information theory studies efficient coding algorithms
  – complexity, compression, likelihood of error

Compression

- There are two main categories
  - 无损 Lossless (Input message = Output message)
  - 有损 Lossy (Input message ≠ Output message)
    - Not necessarily reduce quality (example?)

Compression

$R = \frac{\#\text{ uncompressed}}{\#\text{ compressed}}$

$$\text{Compression Ratio} = \frac{\text{Uncompressed Size}}{\text{Compressed Size}}$$

$$\text{Space Savings} = 1 - \frac{\text{Compressed Size}}{\text{Uncompressed Size}}$$

space savings $= 1 - \frac{1}{R}$

Example

- Compress a 10MB file to 2MB

$R = \frac{10\,MB}{2\,MB} = 5 \rightarrow$ space savings $= 1 - \frac{1}{R} = \frac{4}{5} = 0.8$

- Compression ratio = 5 or 5:1

- Space savings = 0.8 or 80%

Terminology

- Coding (encoding) maps source messages from alphabet (S) into codewords (C)

- Source message (symbol) is basic unit into which a string is partitioned
  - can be a single letter or a string of letters

Terminology (Types)

- Block-block message $\xrightarrow{\text{map}}$ codeword
  - source message and codeword: fixed length
  - e.g., ASCII
- Block-variable
  - source message: fixed; codeword: variable
  - e.g., Huffman coding
- Variable-block
  - source message: variable; codeword: fixed
  - e.g., LZW
- Variable-variable
  - source message and codeword: variable
  - e.g., Arithmetic coding

Terminology (Symmetry)

- Symmetric compression
  - requires same time for encoding and decoding
  - used for live mode applications (teleconference)

- Asymmetric compression 只加密一次但解多次解密
  - performed once when enough time is available
  - decompression performed frequently, must be fast
  - used for retrieval mode applications (e.g., an interactive CD-ROM)

Decodable

- A code is
  - distinct if each codeword can be distinguished from every other (mapping is one-to-one)
  - uniquely decodable if every codeword is identifiable when immersed in a sequence of codewords

Example

- A: 1
- B: 10
- C: 11
- D: 101
- To encode ABCD: 11011101
- To decode 11011101: ?

A B C D

Uniquely decodable

- Uniquely decodable is a prefix free code
  - if no codeword is a proper prefix of any other

- For example {1, 100000, 00} is uniquely decodable, but is not a prefix code
  - consider the codeword {…1000000001…}

- In practice, we prefer prefix code (why?)

Example

S	Code
a	00
b	01
c	10
d	110
e	111

Example

S	Code
a	00
b	01
c	10
d	110
e	111

010001001101100 0

b a b a d d a

⇓

babadda

Example

S	Code
a	00
b	01
c	10
d	110
e	111

0100010011011000

b a b a d d a

Static codes

- Mapping is fixed before transmission
  - E.g., Huffman coding

- probabilities known in advance

Dynamic codes

- Mapping changes over time
  - i.e. adaptive coding
- Attempts to exploit locality of reference
  - periodic, frequent occurrences of messages
  - e.g., dynamic Huffman

Traditional evaluation criteria

- Algorithm complexity
  - running time
- Amount of compression
  - redundancy
  - compression ratio
- How to measure?

entropy → information

$H(X) = \sum_{x \in X} -p(x) \log p(x)$

Measure of information

- Consider symbols s_i and the probability of occurrence of each symbol $p(s_i)$
- In case of fixed-length coding , smallest number of bits per symbol needed is
  - $L \geq \log_2(N)$ bits per symbol
  - Example: Message with 5 symbols need 3 bits ($L \geq \log_2 5$)

Variable length coding

- Also known as entropy coding
  - The number of bits used to code symbols in the alphabet is variable
  - E.g. Huffman coding, Arithmetic coding

Entropy

- What is the minimum number of bits per symbol?
- Answer: Shannon's result – theoretical minimum average number of bits per code word is known as Entropy (H)

$$\sum_{i=1}^{n} - p(s_i) \log_2 p(s_i)$$

Entropy example

- Alphabet S = {A, B}
  - p(A) = 0.4; p(B) = 0.6
- Compute Entropy (H)
  - $-0.4*\log_2 0.4 + -0.6*\log_2 0.6 = .97$ bits
- Maximum uncertainty (gives largest H)
  - occurs when all probabilities are equal

Example: ASCII

0	nul	1	soh	2	stx	3	etx	4	eot	5	enq	6	ack	7	bel
8	bs	9	ht	10	nl	11	vt	12	np	13	cr	14	so	15	si
16	dle	17	dc1	18	dc2	19	dc3	20	dc4	21	nak	22	syn	23	etb
24	can	25	em	26	sub	27	esc	28	fs	29	gs	30	rs	31	us
32	sp	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	del

ASCII

- Example: SPACE is 32 or 00100000. ‘z’ is 122 or 01111010
- 256 symbols, assume <u>same probability</u> for each
- P(s) = 1/256
- Optimal length for each char is log 1/P(s) => log 256 = 8 bits

David A. Huffman

David Huffman is best known for the invention of Huffman code, a highly important compression scheme for lossless variable length encoding. It was the result of a term paper he wrote while a graduate student at the Massachusetts Institute of Technology (MIT)…

From: Wikipedia

Huffman coding algorithm

1. Take the two least probable symbols in the alphabet

   (longest code words, equal length, differing in last digit)

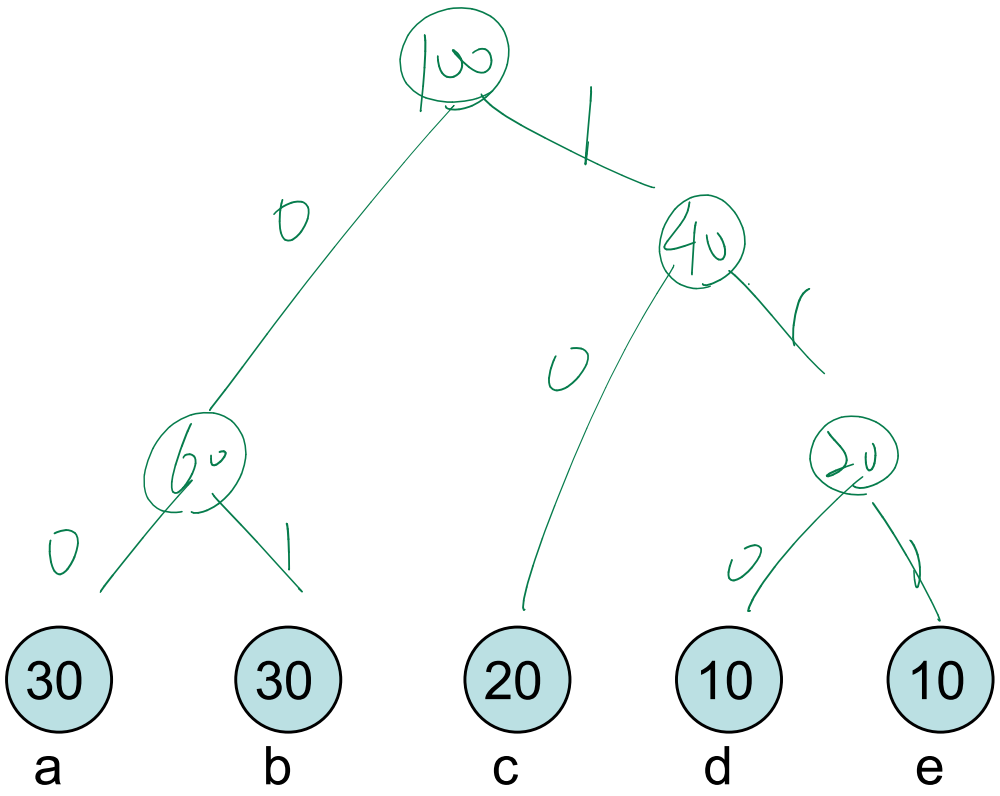
2. Combine these two symbols into a single symbol
3. Repeat

Example: Huffman coding

S	Freq
a	30
b	30
c	20
d	10
e	10

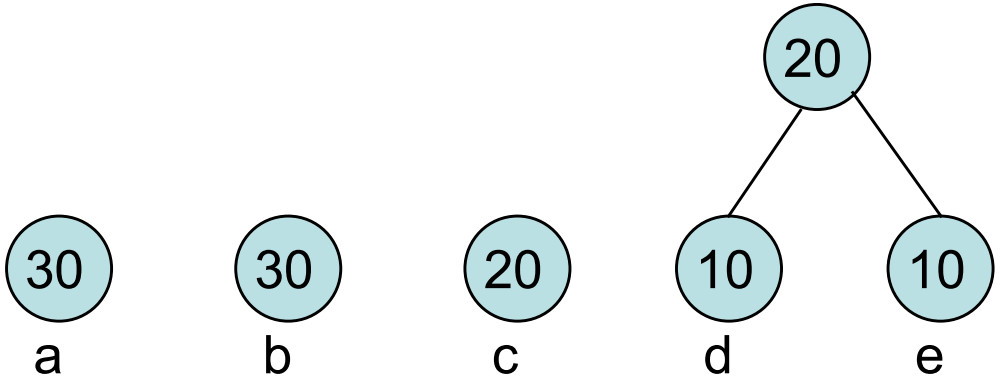
Example

S	Freq	Huffman
a	30	00
b	30	01
c	20	10
d	10	110
e	10	111



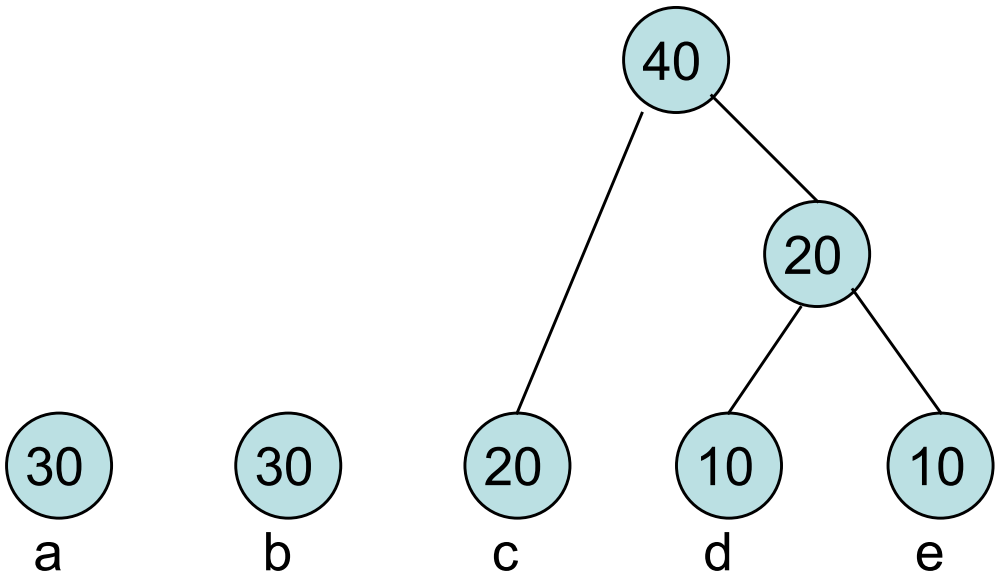
Example

S	Freq	Huffman
a	30	
b	30	
c	20	
d	10	
e	10	



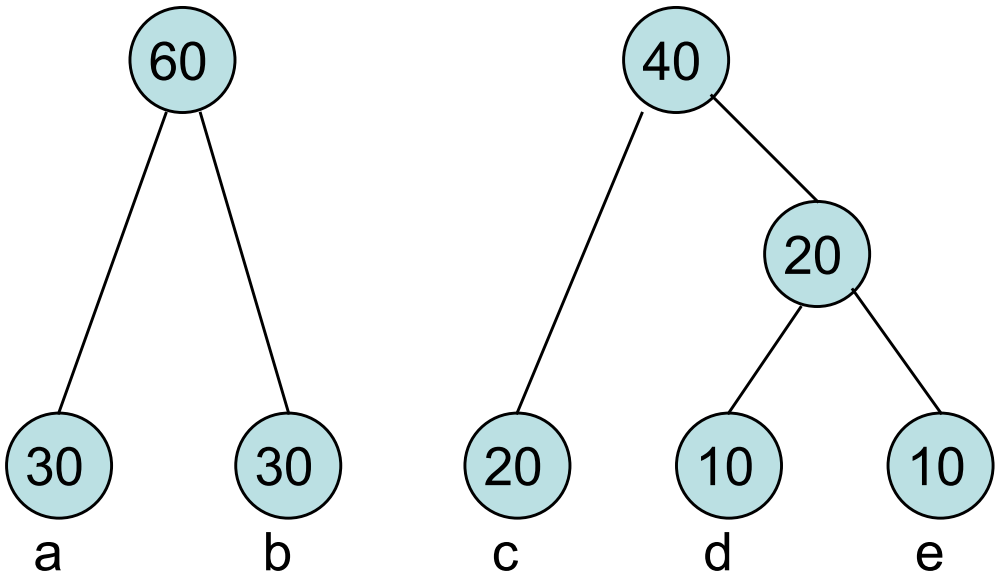
Example

S	Freq	Huffman
a	30	
b	30	
c	20	
d	10	
e	10	



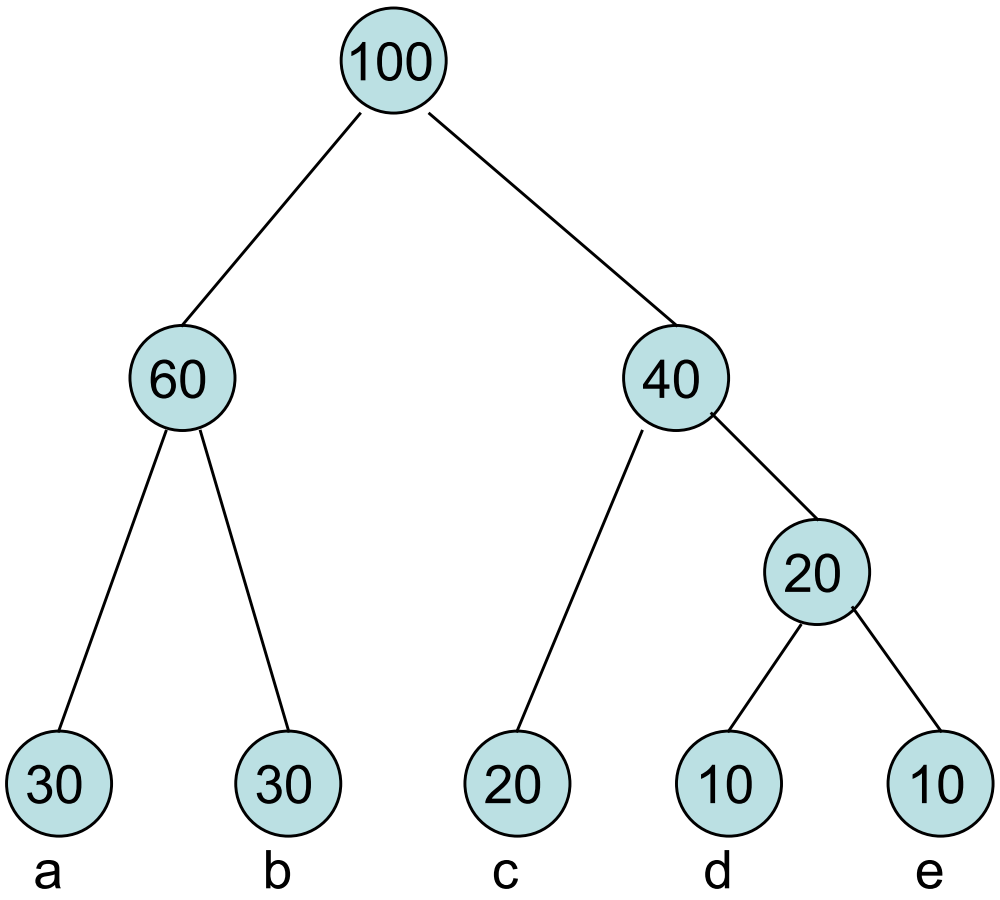
Example

S	Freq	Huffman
a	30	
b	30	
c	20	
d	10	
e	10	



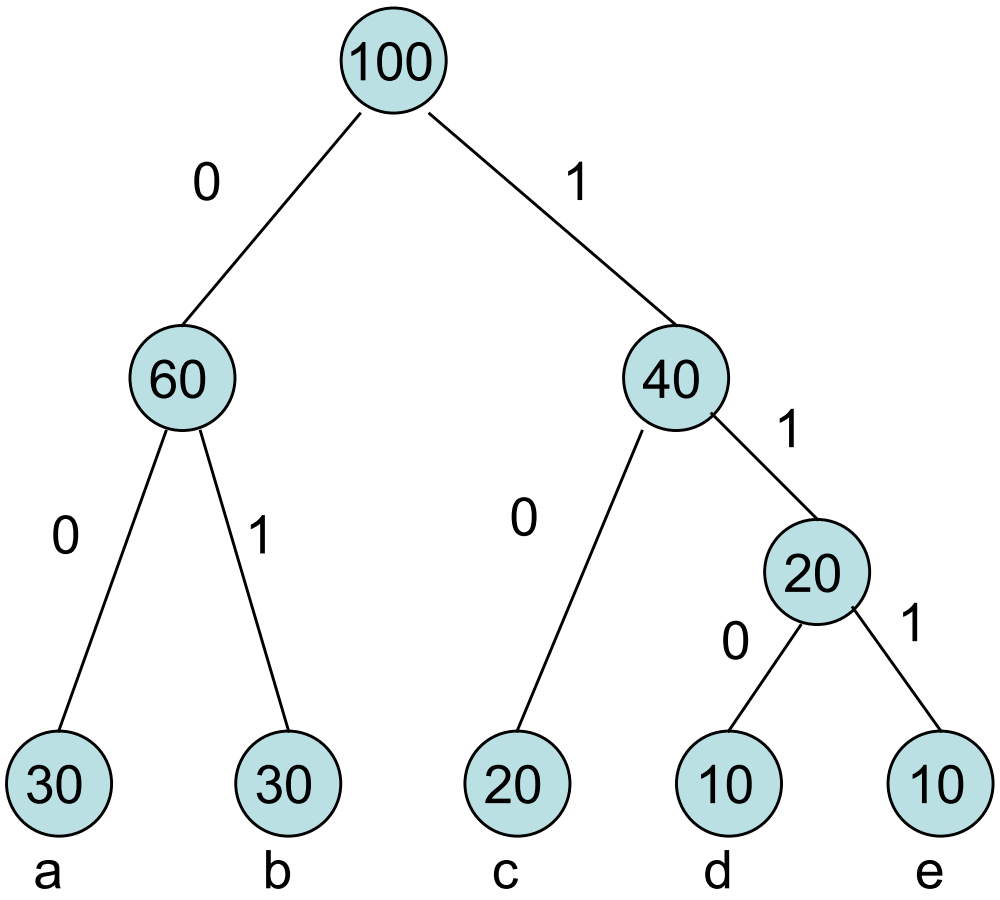
Example

S	Freq	Huffman
a	30	
b	30	
c	20	
d	10	
e	10	



Example

S	Freq	Huffman
a	30	00
b	30	01
c	20	10
d	10	110
e	10	111



Average length L

= (30*2 + 30*2 + 20*2 + 10*3 + 10*3) / 100

= 220 / 100

= <u>2.2</u>

$$H(X) = -\sum_{x \in X} p(x) \log p(x)$$

$$= -(0.3 \log 0.3 + 0.3 \log 0.3 + 0.2 \log 0.2 + 2 \times 0.1 \log 0.1)$$

$$= 2.171 \text{ bits}$$

$$\text{Average Length} = \sum_{x \in X} p(x) \cdot N(x)$$

$$= 0.3 \times 2 + 0.3 \times 2 + 0.2 \times 2 + 0.1 \times 3 + 0.1 \times 3$$

$$= 0.6 + 0.6 + 0.4 + 0.3 + 0.3$$

$$= 2.2 \text{ bits}$$

$\Rightarrow$ Average Length $\geq$ entropy

Average length L

$= (30*2 + 30*2 + 20*2 + 10*3 + 10*3) / 100$

$= 220 / 100$

$= \underline{2.2}$

Better than using fixed length 3 bits for 5 symbols.

Entropy

H = -0.3 * log 0.3 + -0.3 * log 0.3 + -0.2 * log 0.2 + -0.1 * log 0.1 + -0.1 * log 0.1

= -0.3*(-1.737) + -0.3*(-1.737) + -0.2 * (-2.322) + -0.1 * (-3.322) + -0.1 * (-3.322)

= 0.3 log 10/3 + 0.3 log 10/3 + 0.2 log 5 + 0.1 log 10 + 0.1 log 10

= 0.3*1.737 + 0.3*1.737 + 0.2* 2.322 + 0.1*3.322 + 0.1*3.322

= <u>2.17</u>

Another example

- S={a, b, c, d} with freq {4, 2, 1, 1}
- $H = 4/8*\log_2 2 + 2/8*\log_2 4 + 1/8*\log_2 8 + 1/8*\log_2 8$
- $H = 1/2 + 1/2 + 3/8 + 3/8 = 1.75$
- a => 0 b => 10 c => 110 d => 111
- Message: {abcdabaa} => {0 10 110 111 0 10 0 0}
- Average length L = 14 bits / 8 chars = 1.75
- If equal probability, i.e. fixed length, need $\log_2 4 = 2$ bits

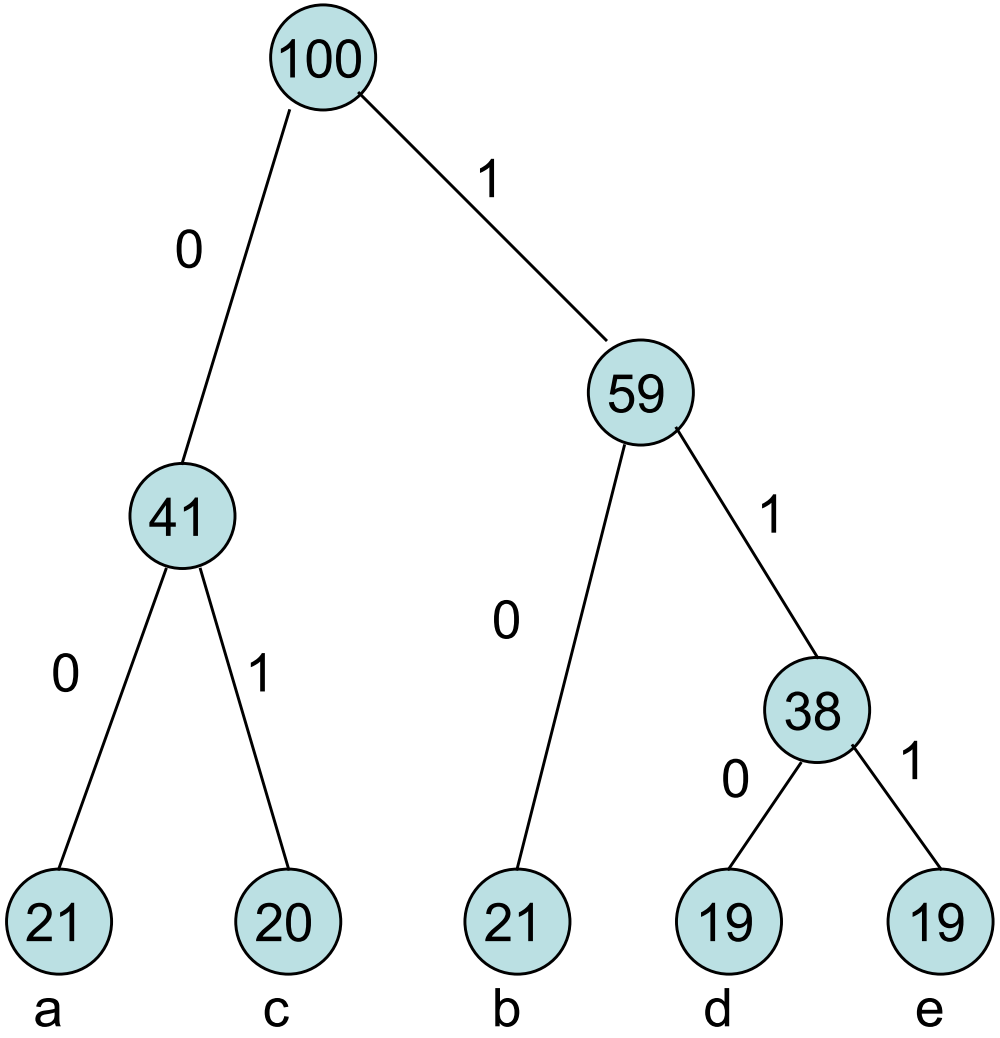
Huffman coding

S	Freq	Huffman
a	~~30~~ 21	
b	~~30~~ 21	
c	~~20~~ 20	
d	~~10~~ 19	
e	~~10~~ 19	

Total: 100

Huffman coding

S	Freq	Huffman
a	21	00
b	21	10
c	20	01
d	19	110
e	19	111



Huffman optimal?

H = 0.21 log 100/21 + 0.21 log 100/21 + 0.2 log 5 + 0.19 log 100/19 + 0.19 log 100/19

= 0.21*2.252 + 0.21*2.252 + 0.2* 2.322 + 0.19*2.396 + 0.19*2.396

= <u>2.32</u>

L = (21*2 + 21*2 + 20*2 + 19*3 + 19*3)/100

= <u>2.38</u>

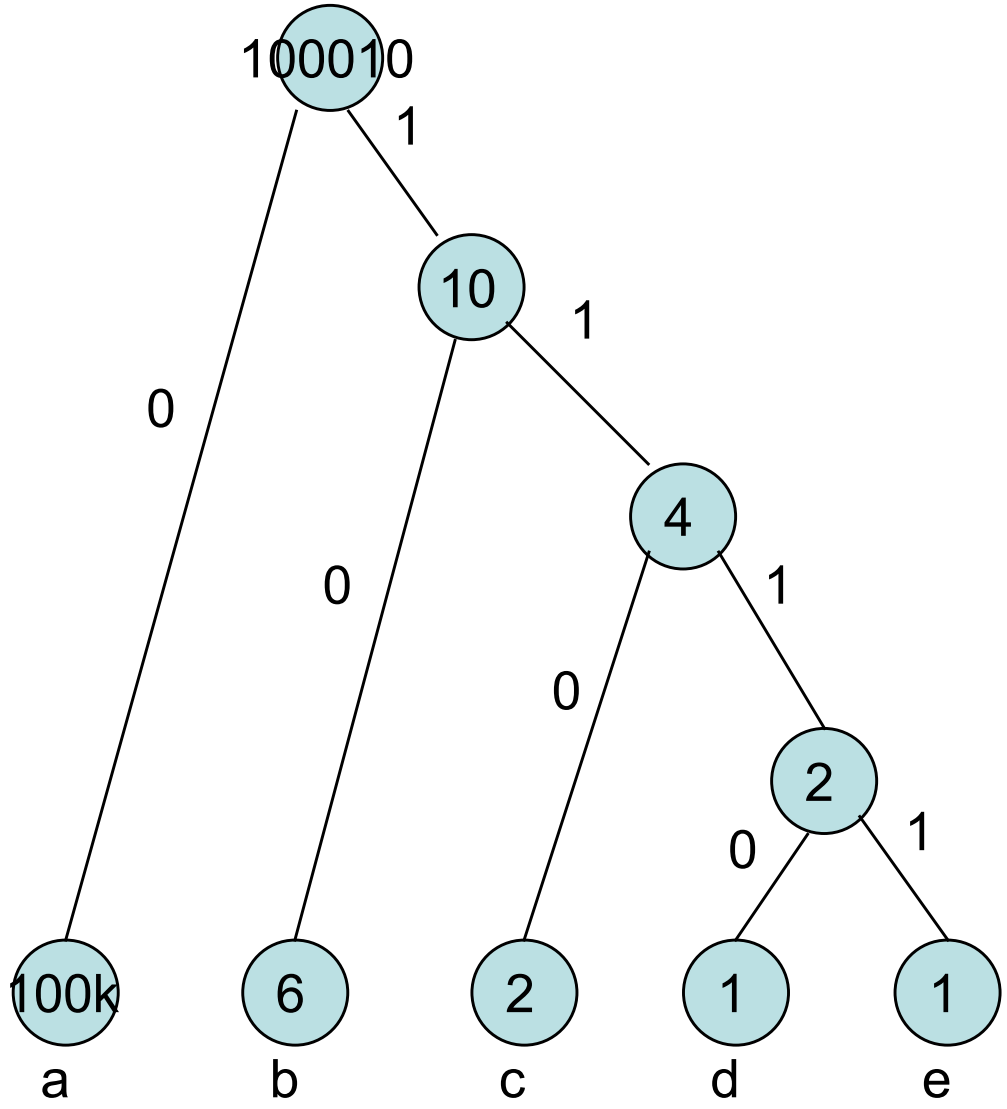
Huffman coding

S	Freq	Huffman
a	~~30~~ 100000	
b	~~30~~ 6	
c	~~20~~ 2	
d	~~10~~ 1	
e	~~10~~ 1	

Total: 100010

Huffman coding

S	Freq	Huffman
a	100000	0
b	6	10
c	2	110
d	1	1110
e	1	1111



Huffman optimal?

$$H = 0.9999 \log 1.0001 + 0.00006 \log 16668.333 + \ldots + 1/100010 \log 100010$$

$$\approx 0.00$$

$$L = (100000*1 + \ldots)/100010$$

$$\approx 1$$

Problems of Huffman coding

- Huffman codes have an integral # of bits.
  - E.g., log (3) = 1.585 while Huffman may need 2 bits
- Noticeable non-optimality when prob of a symbol is high.

Best for average probability.

=> Arithmetic coding

Arithmetic coding

Message to encode:

BILL GATES

Character	Probability
SPACE	1/10
A	1/10
B	1/10
E	1/10
G	1/10
I	1/10
L	2/10
S	1/10
T	1/10

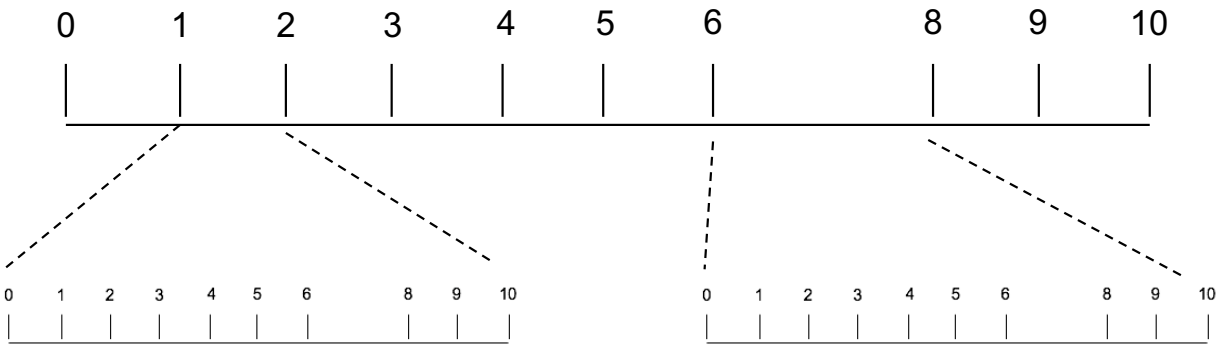
Example extracted from February, 1991 issue of Dr. Dobb's Journal

Arithmetic coding

Character	Probability	Range
SPACE	1/10	0.00 - 0.10
A	1/10	0.10 - 0.20
B	1/10	0.20 - 0.30
E	1/10	0.30 - 0.40
G	1/10	0.40 - 0.50
I	1/10	0.50 - 0.60
L	2/10	0.60 - 0.80
S	1/10	0.80 - 0.90
T	1/10	0.90 - 1.00

Arithmetic coding

Character	Probability	Range
SPACE	1/10	0.00 - 0.10
A	1/10	0.10 - 0.20
B	1/10	0.20 - 0.30
E	1/10	0.30 - 0.40
G	1/10	0.40 - 0.50
I	1/10	0.50 - 0.60
L	2/10	0.60 - 0.80
S	1/10	0.80 - 0.90
T	1/10	0.90 - 1.00



Arithmetic coding algorithm

```
Set low to 0.0
Set high to 1.0
While there are still input symbols do
    get an input symbol
    code_range = high - low.
    high = low + range*high_range(symbol)
    low = low + range*low_range(symbol)
End of While
output low or a number within the range
```

Arithmetic coding

New Character	Low value	High Value
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
SPACE	0.25720	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.2572168
E	0.257216772	0.257216776
S	0.2572167752	0.2572167756

Example

Consider the second L as new char:

code_range = 0.258 – 0.256 = 0.002

high = 0.256 + 0.002*0.8 = 0.2576

low = 0.256 + 0.002*0.6 = 0.2572

code_range = 0.258 - 0.256 = 0.002

high = 0.256 + code_range × high_range = 2.56 + 0.002 × 0.8 = 0.2576

low = 0.256 + code_range × low_range = 2.56 + 0.002 × 0.6 = 0.2572

Decoding algorithm

get encoded number

Do

find symbol whose range straddles the encoded number

output the symbol

range = symbol high value - symbol low value

subtract symbol low value from encoded number

divide encoded number by range

until no more symbols

$$\text{Next} = \frac{0.72167 - 0.6}{0.8 - 0.6} = 0.6083876$$

Arithmetic coding

Encoded Number	Output Symbol	Low	High	Range
0.2572167752	B	0.2	0.3	0.1
0.572167752	I	0.5	0.6	0.1
0.72167752	L	0.6	0.8	0.2
0.6083876	L	0.6	0.8	0.2
0.041938	SPACE	0.0	0.1	0.1
0.41938	G	0.4	0.5	0.1
0.1938	A	0.2	0.3	0.1
0.938	T	0.9	1.0	0.1
0.38	E	0.3	0.4	0.1
0.8	S	0.8	0.9	0.1
0.0				

Example

At the first L, encoded number is 0.72167752.

output the first L

range = 0.8 – 0.6 = 0.2

encoded number = (0.72167752 – 0.6) / 0.2

= 0.6083876

Advantage of arithmetic coding

Assume: A 90% END 10%

To encode: AAAAAAA

New Character	Low value	High Value
	0.0	1.0
A	0.0	0.9
A	0.0	0.81
A	0.0	0.729
A	0.0	0.6561
A	0.0	0.59049
A	0.0	0.531441
A	0.0	0.4782969
END	0.43046721	0.4782969

Advantage of arithmetic coding

Assume: A 90% END 10%

To encode: AAAAAAA

New Character	Low value	High Value
	0.0	1.0
A	0.0	0.9
A	0.0	0.81
A	0.0	0.729
A	0.0	0.6561
A	0.0	0.59049
A	0.0	0.531441
A	0.0	0.4782969
END	0.43046721	0.4782969

e.g., 0.45

Patents on AC

- Bzip2 and JPG use Huffman as AC protected by patents
- PackJPG using AC shows 25% of size saving

Some AC patents (expiring)

U.S. Patent 4,122,440 — (IBM) Filed 4 March 77, Granted 24 October 78 (Now expired)

U.S. Patent 4,286,256 — (IBM) Granted 25 August 81 (Now expired)

U.S. Patent 4,467,317 — (IBM) Granted 21 August 84 (Now expired)

U.S. Patent 4,652,856 — (IBM) Granted 4 February 86 (Now expired)

U.S. Patent 4,891,643 — (IBM) Filed 15 September 86, granted 2 January 90 (Now expired)

U.S. Patent 4,905,297 — (IBM) Filed 18 November 88, granted 27 February 90 (Now expired)

U.S. Patent 4,933,883 — (IBM) Filed 3 May 88, granted 12 June 90 (Now expired)

U.S. Patent 4,935,882 — (IBM) Filed 20 July 88, granted 19 June 90 (Now expired)

U.S. Patent 4,989,000 — Filed 19 June 89, granted 29 January 91 (Now expired)

U.S. Patent 5,099,440 — (IBM) Filed 5 January 90, granted 24 March 92 (Now expired)

U.S. Patent 5,272,478 — (Ricoh) Filed 17 August 92, granted 21 December 93 (Now expired)