COMP9319 Web Data Compression and Search

Course Overview,

Information Representation & Background

Agenda for today

- What is COMP9319

- Information representation
- Compression – a preliminary overview

- What is COMP9319 again
- Should you enrol OR not?

What is COMP9319 ?

Web Data Compression and Search

COMP9319 | **6 Units of Credit**

You are v

Overview

Conditions for Enrolment

Delivery

Course Outline

Fees

Overview

Data Compression: Adaptive Coding, Information Theory; Text Compression (ZIP, GZIP, BZIP, etc); Burrows-Wheeler Transform and Backward Search; XML Compression

Search: Indexing; Pattern Matching and Regular Expression Search; Distributed Querying; Fast Index Construction

The lecture materials will be complemented by projects and assignments.

Read Less

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.

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

Assumed knowledge

Data structures and algorithms: COMP2521 / COMP1927 / COMP9024.

Plus C or C++ programming, e.g.:

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

What is COMP9319 ?

Compression👎

What is COMP9319 ?

Search👌

What is COMP9319 ?

Compression *and* **Search**👍

What is COMP9319 ?

Web Data 🫶

Compression *and* Search👍

Compression

- What (is data compression)
- Why (data compression)
- Where

Compression

- Minimize amount of information to be stored / transmitted

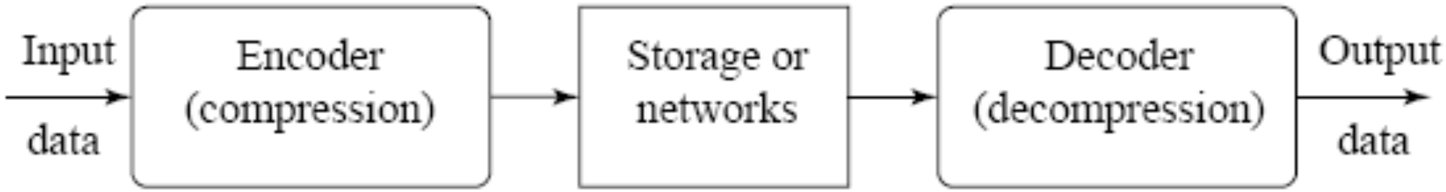
- Transform a sequence of characters into a new bit sequence
  - same information content (for lossless)
  - as short as possible

Compression

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

Compression

- 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

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

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

Example

- Compress a 10MB file to 2MB
- Compression ratio = 5 or 5:1
- Space savings = 0.8 or 80%

Familiar tools

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

Your **first** compression algorithm in COMP9319

raaabbccccdabbbbeee$

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*)

RLE

raaabbccccdaaaaabbbbeeeeeed$

r1a3b2c4d1a5b4e6d1$

RLE

raaabbccccdaaaaabbbbeeeeeed$

r1a3b2c4d1a5b4e6d1$

Too simple?

RLE

raaabbccccdaaaaabbbbeeeeeed$

r1a3b2c4d1a5b4e6d1$

ra3bbc4da5b4e6d$

RLE

raaabbccccdaaaaabbbbeeeeeed$

r1a3b2c4d1a5b4e6d1$

ra3bbc4da5b4e6d$

ra0bbc1da2b1e3d$

Problem: runs are usually ”small”

rabcabcababaabacabcabcabcababaa$

A glimpse of BWT

rabcabcababaabacabcabcabcababaa$

aabbbbccacccrcbaaaaaaaaaabbbbba$

BWT+RLE

rabcabcababaabacabcabcabcababaa$

aabbbbccacccrcbaaaaaaaaaabbbbba$

aab4ccac3rcba10b5a$

Compression? Where?

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"

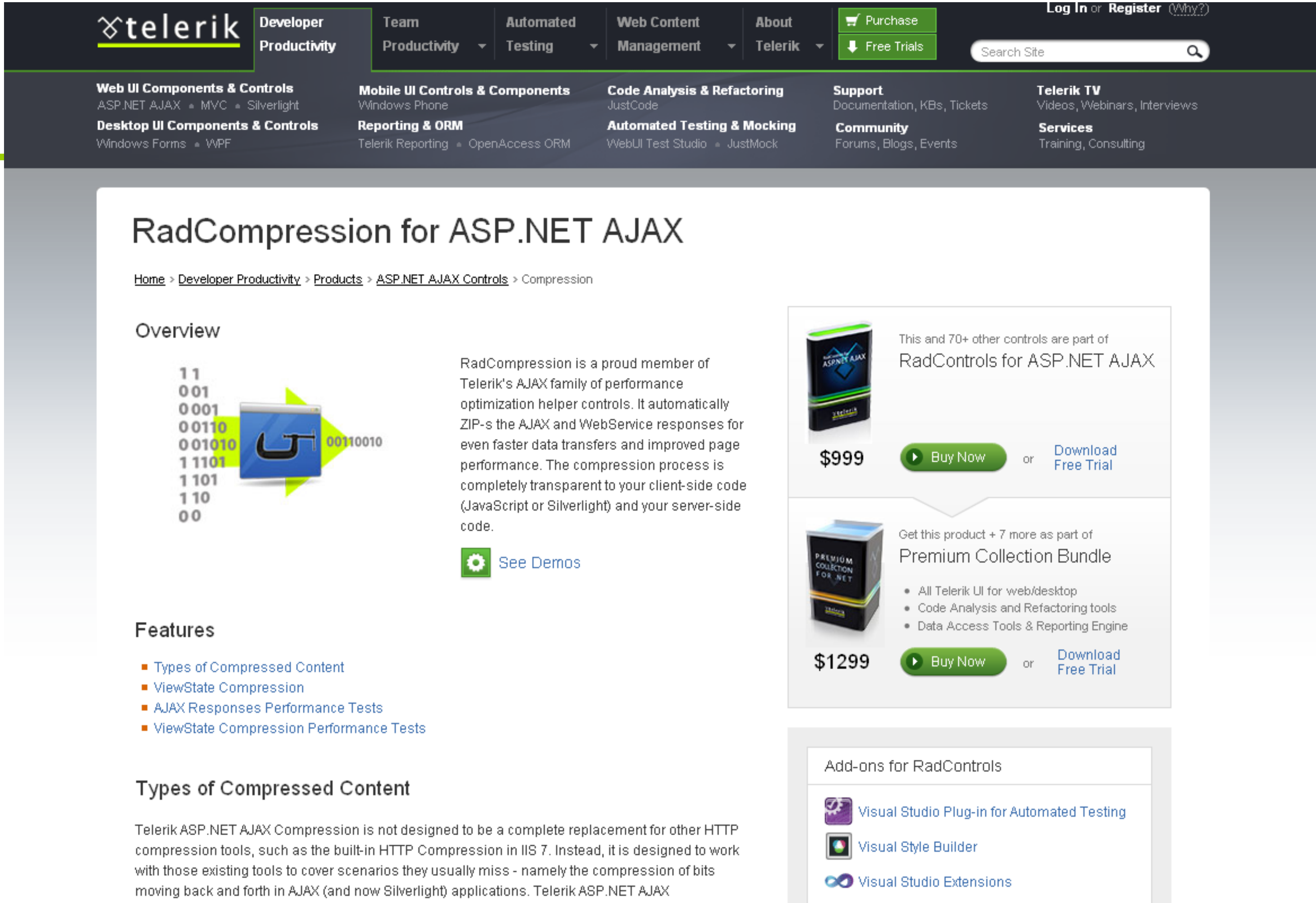
Accept-Ranges: bytes

Content-Length: 438

Connection: close

Content-Type: text/html; charset=UTF-8

Content-Encoding: gzip


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
Features
Types of Compressed Content
ViewState Compression
AJAX Responses Performance Tests
ViewState Compression Performance Tests
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
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
Add-ons for RadControls
Visual Studio Plug-in for Automated Testing
Visual Style Builder
Visual Studio Extensions

Storwize

Better Storage Utilization

Reduces existing storage utilization up to 80%

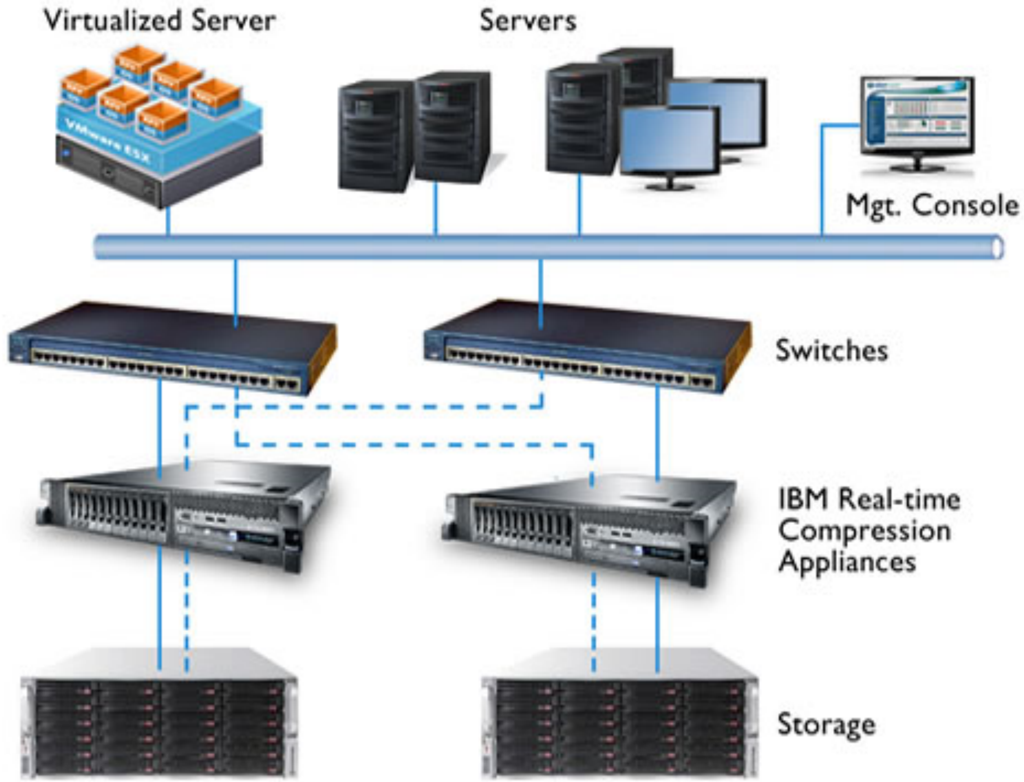
No performance degradation

Lowers Capital and Operational Costs

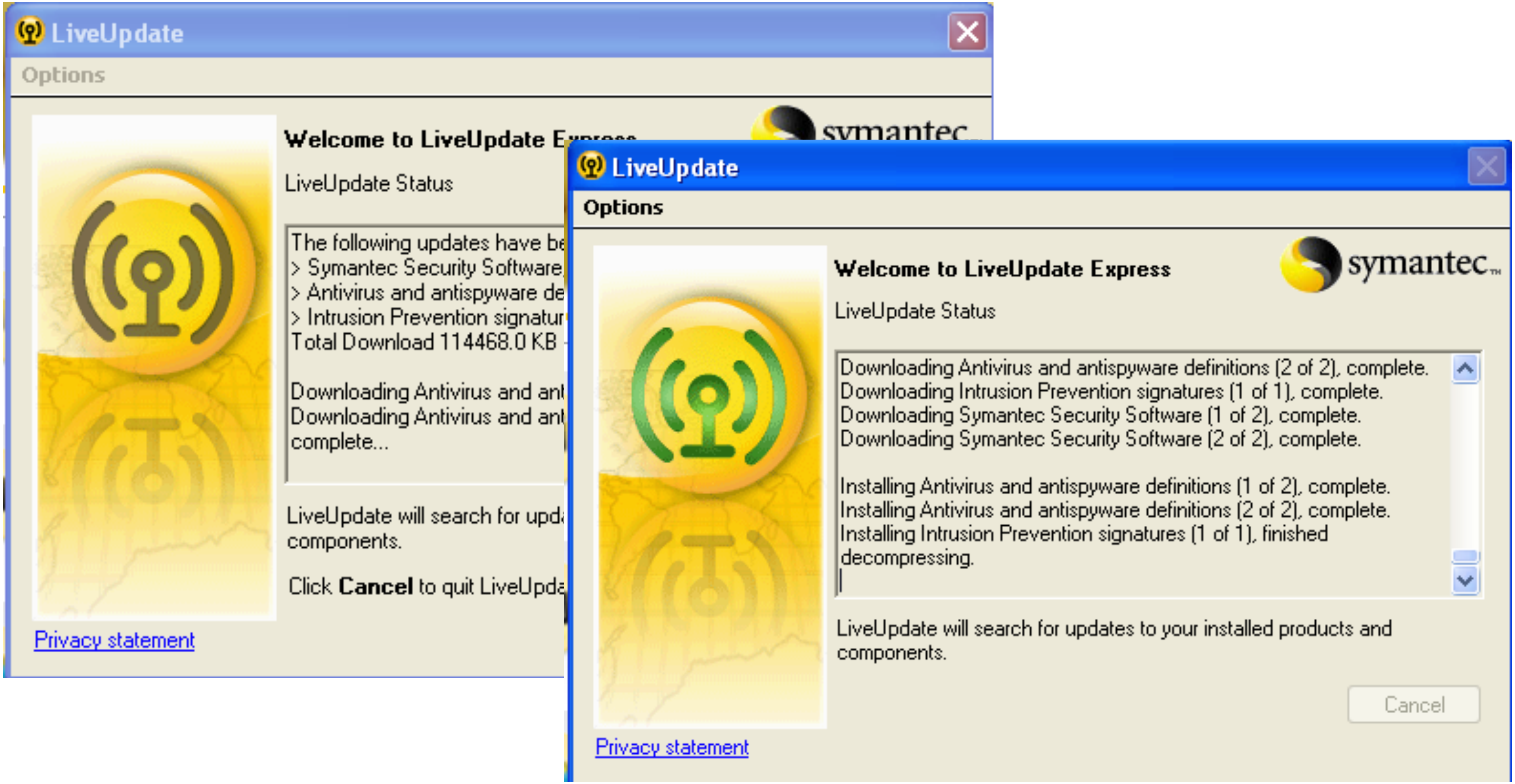
Better Energy Efficiency

Less to store, power and cool

…



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/Web services

  e.g., JSON

Big techs

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$
```

Compression for non-compression applications: e.g., 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

More examples

Login to a CSE Linux machine and then:

```
cs9319@vx11:~$ cd ~cs9319/wk1
cs9319@vx11:~/wk1$ ls
bibrec.bwt   eg1-lossy.rle   eg2.bin-prob1   readme-eg1.rle-bin
bibrec.txt   eg1.rle         eg2.bin-prob2   readme-eg2.bin-prob1
eg1.bin      eg1.rle-bin     eg2.rle         readme-eg2.bin-prob2
eg1.long1    eg1.txt         eg2.rle-bin     readme-eg2.rle-bin
eg1.long2    eg2.bin         eg2.txt
cs9319@vx11:~/wk1$
```

Example 1: 80 days weather

 S

 R

All sunny days except the last 16 days:

SSS…RRRSSSSSRRRRSSSS

Capture the information

```
cs9319@vx11:~/wk1$ more eg1.long1
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Rainy Day, Rainy Day, Rainy Day, Sunny Day, Sunny Day, Sunny Day,
Sunny Day, Sunny Day, Rainy Day, Rainy Day, Rainy Day, Rainy Day, Sunny Day,
Sunny Day, Sunny Day, Sunny Day
cs9319@vx11:~/wk1$ more eg1.long2
Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny
,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunn
y,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sun
ny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Su
nny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,R
ainy,Rainy,Rainy,Sunny,Sunny,Sunny,Sunny,Sunny,Rainy,Rainy,Rainy,Rainy,Sunny,
Sunny,Sunny,Sunny
```

More efficient representation

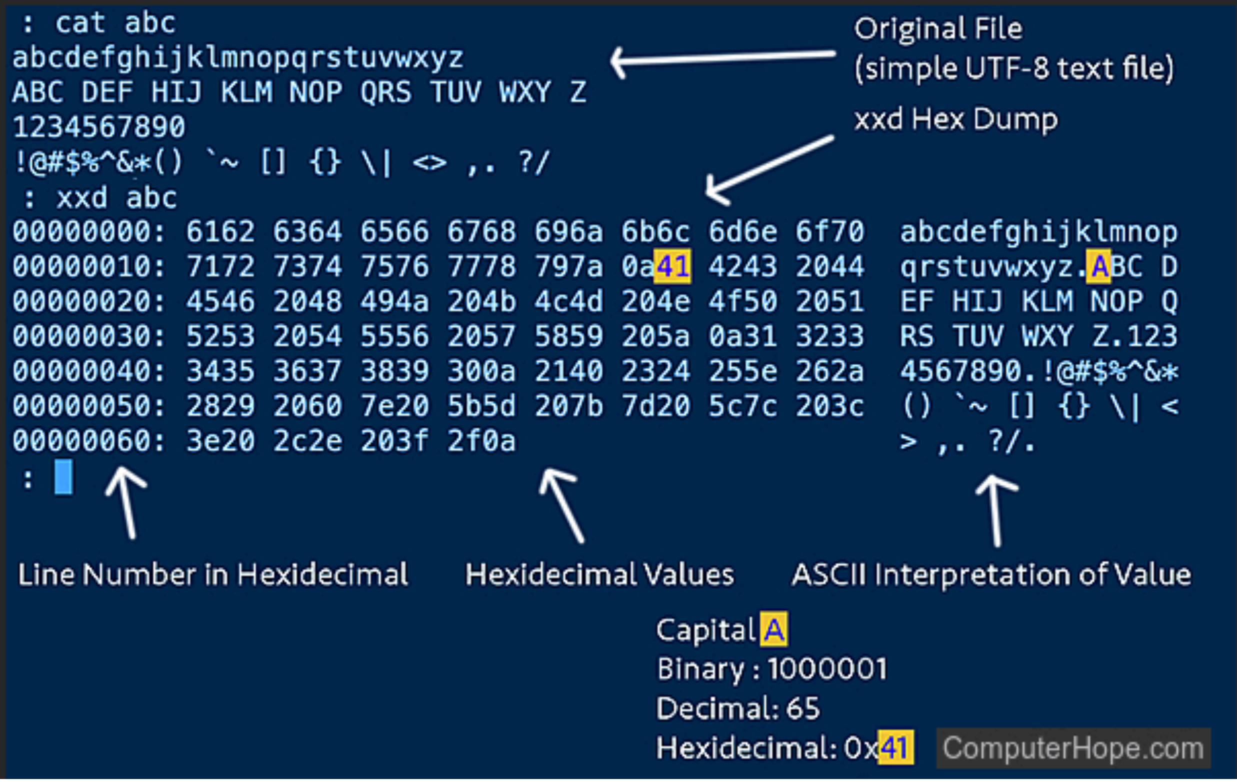
```
cs9319@vx11:~/wk1$ more eg1.long2
Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny
,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunn
y,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sun
ny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Su
nny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,Sunny,R
ainy,Rainy,Rainy,Sunny,Sunny,Sunny,Sunny,Sunny,Rainy,Rainy,Rainy,Rainy,Sunny,
Sunny,Sunny,Sunny
cs9319@vx11:~/wk1$
cs9319@vx11:~/wk1$ more eg1.txt
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSRRRSSSSSRRRRS
SSS
cs9319@vx11:~/wk1$
cs9319@vx11:~/wk1$ ls -l eg1.long* eg1.txt
-rw-r----- 1 cs9319 cs9319 879 May 29 21:37 eg1.long1
-rw-r----- 1 cs9319 cs9319 479 May 29 21:37 eg1.long2
-rw-r----- 1 cs9319 cs9319  80 May 29 21:37 eg1.txt
cs9319@vx11:~/wk1$
```

Even more efficient?

- In binary form?
- Hard to read - xxd is your friend

What is ”xxd”

The **xxd** program takes a file or standard input and outputs a hex dump that uses only ASCII or EBCDIC characters. This output hex dump can be safely emailed and reconstructed at the destination. It can also take an equivalently formatted hex dump and convert it back to binary form, allowing binary files to be edited or patched as text.



Even more efficient?

```
cs9319@vx11:~/wk1$
cs9319@vx11:~/wk1$ more eg1.txt
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSRRRSSSSSRRRRS
SSS
cs9319@vx11:~/wk1$ xxd -b eg1.bin
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 11100000 11110000                    ....
cs9319@vx11:~/wk1$
```

Even even more efficient?

```
cs9319@vx11:~/wk1$ more eg1.txt
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSRRRSSSSSRRRRS
SSS
cs9319@vx11:~/wk1$ more eg1.rle
S63R2S4R3S3
cs9319@vx11:~/wk1$ xxd -b eg1.bin
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 11100000 11110000                    ....
cs9319@vx11:~/wk1$ more readme-eg1.rle-bin
eg1.rle-bin:
1 bit for S or R
7 bit for integer upto 127

cs9319@vx11:~/wk1$ xxd -b eg1.rle-bin
00000000: 00111111 10000010 00000100 10000011 00000011           ?....
cs9319@vx11:~/wk1$ ls -l eg1.txt eg1.rle eg1.bin eg1.rle-bin
-rw-r----- 1 cs9319 cs9319 10 May 29 21:37 eg1.bin
-rw-r----- 1 cs9319 cs9319 11 May 29 21:37 eg1.rle
-rw-r----- 1 cs9319 cs9319  5 May 29 21:37 eg1.rle-bin
-rw-r----- 1 cs9319 cs9319 80 May 29 21:37 eg1.txt
cs9319@vx11:~/wk1$
```

Even even even more efficient?

Even0 more efficient?

Well, if it's okay to lose something:

```
cs9319@vx11:~/wk1$ more eg1.rle
S63R2S4R3S3
cs9319@vx11:~/wk1$ more eg1-lossy.rle
S80
cs9319@vx11:~/wk1$ 
```

Example 2: 80 days weather

 S  C

 R

All sunny days except the last 16 days:

SSS…RRRCSSSSRRRRCSSS

We have 3 states instead of 2

We know ”binary” is better:

```
cs9319@vx11:~/wk1$ more eg2.txt
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSRRRCSSSSRRRRC
SSS
cs9319@vx11:~/wk1$ xxd -b eg2.bin
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 00000000 00000000 00000000 00000000  ......
0000000c: 00000000 00000000 00000000 00000000 01010110 00000000  ....V.
00000012: 01010101 10000000                                      U.
```

States are not of equal prob

Note: We do not like equal probability !!!

WHY?

States are not of equal prob

Note: We do not like equal probability !!!

```
cs9319@vx11:~/wk1$ more eg2.txt
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSRRRCSSSSRRRRC
SSS
cs9319@vx11:~/wk1$ xxd -b eg2.bin
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 00000000 00000000 00000000 00000000  ......
0000000c: 00000000 00000000 00000000 00000000 01010110 00000000  ....V.
00000012: 01010101 10000000                                      U.
cs9319@vx11:~/wk1$ more readme-eg2.bin-prob1
prob1:
S - 0
R - 10
C - 11

cs9319@vx11:~/wk1$ xxd -b eg2.bin-prob1
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 10101011 00001010 10101100 00000000  ......
cs9319@vx11:~/wk1$
```

ASCII Code

Unfortunately: ASCII assumes equal probability!

```
cs9319@vx11:~/wk1$ ascii -d
    0 NUL    16 DLE    32      48 0    64 @    80 P    96 `   112 p
    1 SOH    17 DC1    33 !    49 1    65 A    81 Q    97 a   113 q
    2 STX    18 DC2    34 "    50 2    66 B    82 R    98 b   114 r
    3 ETX    19 DC3    35 #    51 3    67 C    83 S    99 c   115 s
    4 EOT    20 DC4    36 $    52 4    68 D    84 T   100 d   116 t
    5 ENQ    21 NAK    37 %    53 5    69 E    85 U   101 e   117 u
    6 ACK    22 SYN    38 &    54 6    70 F    86 V   102 f   118 v
    7 BEL    23 ETB    39 '    55 7    71 G    87 W   103 g   119 w
    8 BS     24 CAN    40 (    56 8    72 H    88 X   104 h   120 x
    9 HT     25 EM     41 )    57 9    73 I    89 Y   105 i   121 y
   10 LF     26 SUB    42 *    58 :    74 J    90 Z   106 j   122 z
   11 VT     27 ESC    43 +    59 ;    75 K    91 [   107 k   123 {
   12 FF     28 FS     44 ,    60 <    76 L    92 \   108 l   124 |
   13 CR     29 GS     45 -    61 =    77 M    93 ]   109 m   125 }
   14 SO     30 RS     46 .    62 >    78 N    94 ^   110 n   126 ~
   15 SI     31 US     47 /    63 ?    79 O    95 _   111 o   127 DEL
cs9319@vx11:~/wk1$
```

UTF8

Fortunately

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	[b]U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

States are not of equal prob

There is a minor issue below though:

```
cs9319@vx11:~/wk1$ more eg2.txt
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSRRRCSSSSRRRRC
SSS
cs9319@vx11:~/wk1$ xxd -b eg2.bin
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 00000000 00000000 00000000 00000000  ......
0000000c: 00000000 00000000 00000000 00000000 01010110 00000000  ....V.
00000012: 01010101 10000000                                      U.
cs9319@vx11:~/wk1$ more readme-eg2.bin-prob1
prob1:
S - 0
R - 10
C - 11

cs9319@vx11:~/wk1$ xxd -b eg2.bin-prob1
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 10101011 00001010 10101100 00000000  ......
cs9319@vx11:~/wk1$
```

States are not of equal prob

Problem solved:

```
cs9319@vx11:~/wk1$ more readme-eg2.bin-prob1
prob1:
S - 0
R - 10
C - 11

cs9319@vx11:~/wk1$ xxd -b eg2.bin-prob1
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 10101011 00001010 10101100 00000000  ......
cs9319@vx11:~/wk1$ more readme-eg2.bin-prob2
prob2:
S - 0
R - 10
C - 110
EOF - 111

cs9319@vx11:~/wk1$ xxd -b eg2.bin-prob2
00000000: 00000000 00000000 00000000 00000000 00000000 00000000  ......
00000006: 00000000 00000000 10101011 00000101 01010110 00011100  ....V.
cs9319@vx11:~/wk1$
```

Even more efficient?

Yes, RLE again!

```
cs9319@vx11:~/wk1$ more eg2.txt
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSRRRCSSSSRRRRC
SSS
cs9319@vx11:~/wk1$ more eg2.rle
S63R2CS3R3CS2
cs9319@vx11:~/wk1$ more readme-eg2.rle-bin
eg2.rle-bin:
2 bit for S, R or C
6 bit for integer upto 63

cs9319@vx11:~/wk1$ xxd -b eg2.rle-bin
00000000: 00111111 10000010 11000000 00000011 10000011 11000000  ?.....
00000006: 00000010                                               .
cs9319@vx11:~/wk1$ ls -l eg2.*
-rw-r----- 1 cs9319 cs9319 20 May 29 21:37 eg2.bin
-rw-r----- 1 cs9319 cs9319 12 May 29 21:37 eg2.bin-prob1
-rw-r----- 1 cs9319 cs9319 12 May 29 21:37 eg2.bin-prob2
-rw-r----- 1 cs9319 cs9319 13 May 29 21:37 eg2.rle
-rw-r----- 1 cs9319 cs9319  7 May 29 21:37 eg2.rle-bin
-rw-r----- 1 cs9319 cs9319 80 May 29 21:37 eg2.txt
cs9319@vx11:~/wk1$
```

Problem of RLE

Useful documents don’t usually have runs (rarely have a continuous sequence of the same character).

bibrec.txt

```
port, San Jose, California[345]November[348]1971[351]RJ935[356]Markus Casper[
359]Gayane Grigoryan[362]Oliver Gronz[365]Oliver Gutjahr[368]Gnther Heinemann
[371]Rita Ley[374]Andreas Rock[377]Analysis of projected hydrological behavio
r of catchments based on signature indices[380]Hydrology and Earth System Sci
ences[383]16[386]409-421[389]2012[392]http://dx.doi.org/10.5194/hess-16-409-2
012[395]http://dx.doi.org/10.5194/hess-16-409-2012[400]Klaus Jansen[403]One S
trike Against the Min-Max Degree Triangulation Problem[406]Universitt Trier,
Mathematik/Informatik, Forschungsbericht[409]92-14[412]1992[417]Manfred Laume
n[420]Structured PSB-Update for Optimal Shape Design Problems[423]Universitt
Trier, Mathematik/Informatik, Forschungsbericht[426]96-17[429]1996[434]Reiner
 Horst[437]Nguyen V. Thoai[440]An Integer Concave Minimization Approach for t
he Minimum Concave Cost Capacitated Flow Problem on Networks[443]Universitt T
rier, Mathematik/Informatik, Forschungsbericht[446]94-13[449]1994[454]Reiner
Horst[457]L. D. Muu[460]Nguyen V. Thoai[463]A Decomposition Algorithm for Opt
imization over Efficient Sets[466]Universitt Trier, Mathematik/Informatik, Fo
rschungsbericht[469]97-04[472]1997[477]Christoph Meinel[480]Anna Slobodov[483
]A Unifying Theoretical Background for Some BDD-based Data Structures[486]Uni
versitt Trier, Mathematik/Informatik, Forschungsbericht[489]94-17[492]1994[49
7]Klaus Jansen[500]On the Complexity of a Licence Constrained Job Assignment
Problem[503]Universitt Trier, Mathematik/Informatik, Forschungsbericht[506]92
```

bibrec.bwt

```
ssssssssssneenngnnineiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiirri0000000000000000000
0000000000000000iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiifffiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiyyiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiiieeggrrnnnnneDrtrllff ehh55555555555adiiEeddmmoffiiiiiiiiii
iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiiiuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu
uuuuuuuuuuuuuuuuuuuuuiiiiiiiiinndddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
```

Even gzip benefits from BWT

```
cs9319@vx11:~/wk1$ ls -l bib*
-rw-r----- 1 cs9319 cs9319 1055718 May 29 21:37 bibrec.bwt
-rw-r----- 1 cs9319 cs9319 1055718 May 29 21:37 bibrec.txt
cs9319@vx11:~/wk1$ gzip bib*
cs9319@vx11:~/wk1$ ls -l bib*
-rw-r----- 1 cs9319 cs9319 293504 May 29 21:37 bibrec.bwt.gz
-rw-r----- 1 cs9319 cs9319 340246 May 29 21:37 bibrec.txt.gz
cs9319@vx11:~/wk1$
```

Efficient search

Even better, in the format below, we can search for all “San Jose” matches in constant time independent of the size of the file !!

```
ssssssssssneenngnnineiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiirri0000000000000000000
0000000000000000iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiifffiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiyyiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiiieeggrrnnnnneDrtrllff ehh55555555555adiiEeddmmoffiiiiiiiiii
iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiiiuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu
uuuuuuuuuuuuuuuuuuuuuiiiiiiiiiinnddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
dddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
```

What is COMP9319 ?

- **how** different compression tools work.
- **how** to manage a large amount of text data (on **small** devices or **large** servers).
- **how** to search gigabytes, terabytes or petabytes of data.
- **how** to perform full text search efficiently with heavy indexing, light indexing / no indexing.
- optional advanced topics (if time allows): distributed repositories, cloud etc.

Course info

- Course homepage: www.cse.unsw.edu.au/~cs9319
- Email:

  cs9319@cse.unsw.edu.au
- Lectures:
  - Weeks 1**-5, 7-10 (*flexi week 6: no classes*)**
  - Fri 9:00-11:00am (wk1, 2, 10 - *in-person lectures, 1-2hr depending on the topics/your participations/Q&A*)
  - Fri 9:00-11:00am (wk3-5, 7-9 - *online via Collaborate, 1-2hr depending on the topics/your participations/Q&A*)
  - **Approx 3 hrs every week (*pre-recorded lectures*)**

Lecturer in charge

Me: Raymond Wong

Areas: DB/IR/BigD Systems; Text Mining/NLP

Office: K17 Level 2 (Room 213)

Email: wong@cse.unsw.edu.au

Ph: 90659925

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

For **individual** COMP9319 matters, please email:
cs9319@cse.unsw.edu.au

for quicker responses.

Lectures

For 2024T2:

- Live lecture on Friday: 9-11am (1-2hrs)
  - In-person: wk 1, 2, 10
  - Online: wk 3-5, 7-9 (Blackboard Collaborate)
- Pre-recorded topic-based lectures: approx. 3hrs / week (to be released on Monday, except week 1).

•*Live lectures shall be recorded as well (in Moodle's Echo360) – hopefully no glitches*

Recorded Topic-based Lectures

- 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
- Most importantly, you can play them at 1.5x, 2.0x, or replay any subsection
- **Note: we assume that you will watch the recordings every week; and attend & ask any questions at the "<u>consultations</u>" (or at the live lectures).**

Live lectures

- Topic-based recordings are good but lack of an overall picture and no interactions or Q&A
- Hence, there is a 1-hour live lecture (Fri 9am, may go slightly overtime up to 2hr if needed) to give an overview of topics; and go through some "practical" discussions / more examples, and/or answer any Q&A for the topics covered from the prev wk.
- To get the most out of COMP9319, highly recommended to attend.

Exercises

Exercises will be provided on WebCMS regularly (from week 2/3 onwards).

- Brief solutions will be released one week later.
- If you're stuck, please join a consultation. We'll go through steps/approaches.

No tutorials/labs

- Have consultation slots instead
- Specific lecture / exercise / assignment questions can be addressed better in consultations
- Important Q&A will be discussed in the live lectures
- <u>Don't leave your questions till very late</u>, we won't be able to address questions that stacked for many weeks

Consultations

- Week 2 - **Week 11** (excluding wk 6)
- 2 days a week
  - 1 in-person + 1 online
  - Check WebCMS for time & location
  - ➔ So please utilize them.
- Run in a hybrid consultation/tut style
- Q&A for exercises & **assignments**
- Q&A for lecture materials

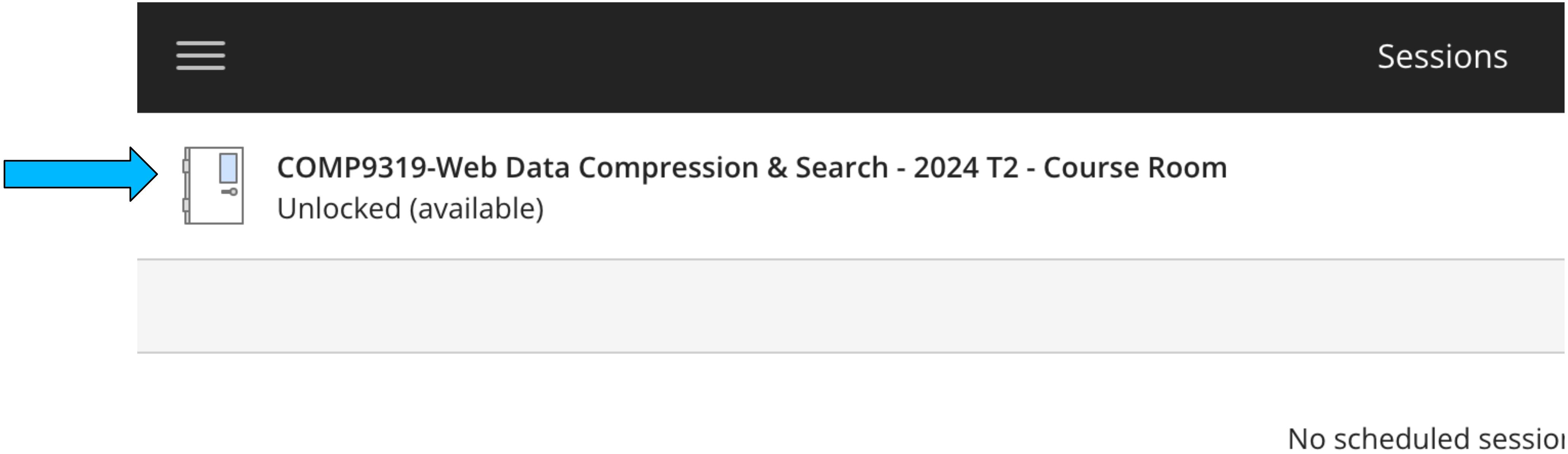
Consultations

Consultations

Day	Start Time	End Time	Room	Who
Tuesday	13:00	14:00	Business School 130 (K-E12-130)	Yan Kin Chi
	In-person consulation. Week 2-5, 7-11.			
Friday	14:00	15:00	Online	Yan Kin Chi
	Online via Blackboard Collaborate (click Course Room, as described in Live Lecture 1). Week 2-5, 7-11.			

Consultations / Live lectures (online)

Join the consultation / live lectures using Course Room in Blackboard Collaborate



The Ed Forum

- For **short** questions only (such as clarification of assignment spec or lecture materials)
- Your peers, tutors, or myself can help answer
- Those often help answer & have good answers may have bonus marks.
- For longer questions, better drop by a consultation session.
- Please do check the Forum regularly for announcements & Q&A

Readings

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

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      = 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 (in-person)

- One final exam (worth 50 %)
- If you are ill on the day of the exam, do not attend the exam – c.f. fit-to-sit policy. Apply for special consideration asap.
- It'll be an **in-person** exam. More details to be provided later in the course.

Two assignments

- 1 smaller prog assignment (15%)
- 1 larger prog assignment (35%)
- Late submission: 5% of the max subtracted from earned marks per day (no acceptance after 5 days late) – *see Course Outline in WebCMS for details.*
-

Programming assignments

- The 1st assignment is *relatively* easier, a warm-up
- The 2nd assignment is larger in scale, and more 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%

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

0. Information Representation (Week 1)

1. Compression

2. Search

3. Compression + Search on plain text

4. “Compression + Search” on Web text

5. Selected advanced topics (if time allows)

In the past, students didn't do well because:

- *Plagiarism*
- Late submission
- Code failed to compile on specified machines
- Program did not follow the spec, i.e., failed most auto-marking

Please **do not** enrol if you

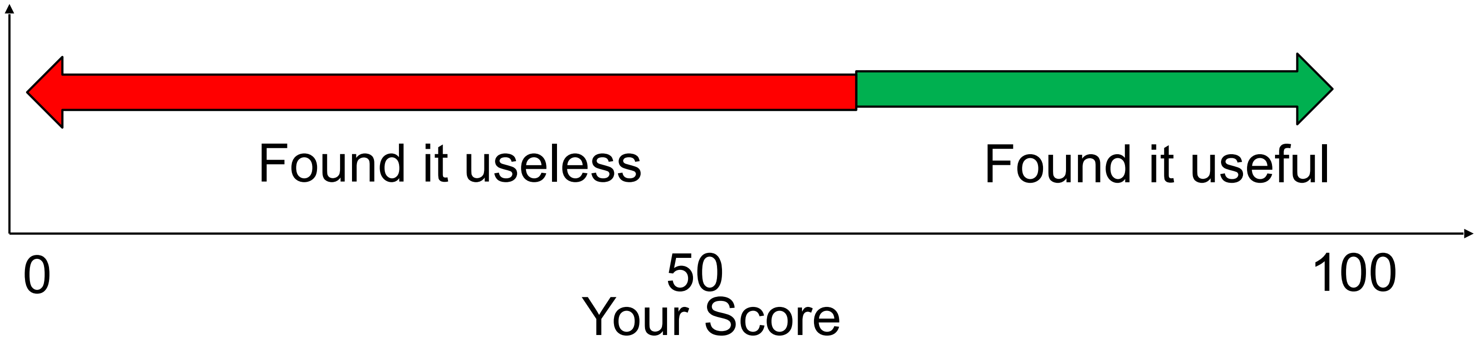
- **Don’t like the setup of COMP9319 (e.g., no tuts, auto-marking for assigts)**
- Not comfortable with COMP2521 / COMP1927 / COMP9024
- Cannot produce correct C/C++ program **on your own**
- Have poor time management
- Are too busy to attend lectures

*It’s important to **READ the Course Outline** on WebCMS before you enrol.*

Is COMP9319 useful ?

It depends on:

1. Your course performance



2. Your field

- Useful in many tech companies / startups such as Google, Amazon
- Not useful for IT applications that system/application performance/scalability is not their priority

QUESTIONS?

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
  - 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
  - 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: ?

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

0100010011011000

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?

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