Zip-Ada

Recent developments in Zip-Ada

Part 1: Overview; new Deflate compression algorithm

Part 2: New LZMA compression algorithm

Dr Gautier de Montmollin

Swiss Ada Event 2017, Rapperswil, September 21, 2017

Overview - what is Zip-Ada ?

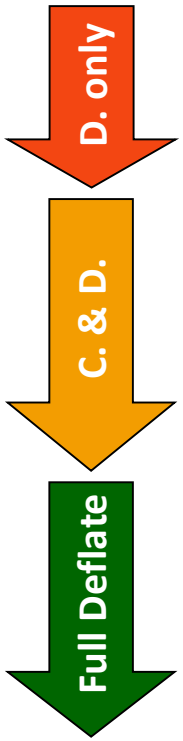
- Zip-Ada is a programming library for dealing with the Zip compressed archive file format (extraction + creation).
- The full sources of Zip-Ada are in Ada – single source set (no “#ifdef” or other conditional precompiler magic) ⇒ limitless portability.
- No interfacing needed
- No worries with linker formats, 32 vs. 64 bits, etc.

- Processing flow and memory footprint can be monitored with the same Ada toolset as the rest of a project.
- Ada streams and exceptions.

OS	CPU	Compiler
OpenVMS	Intel Itanium (64 bit)	GNU - **GNAT**
AIX	Power7 (64 bit)	
MS Windows 9x;NT,2K,XP+	Intel x86 (32 bit)	
MS Windows x64	Intel x64 (64 bit)	
Linux	Intel x86 (32 bit)	
Linux	Intel x86_64 (64 bit)	
Linux on PS3	Cell (64 bit)	
Linux on Raspberry Pi	ARM	
Mac OS X	PowerPC (64 bit)	
Mac OS X	Intel x64 (64 bit)	
Solaris	SPARC (32 or 64 bit)	
Solaris	Intel x64 (64 bit)	
MS-DOS, DR-DOS (Novell)	Intel x86 (16/32 bit)	
OpenBSD	(one of several)	
FreeBSD	Intel x86 (32 bit)	
FreeBSD	Intel x64 (64 bit)	
Android 2.3+	ARM	
MS Windows 9x;NT,2K,XP+	Intel x86 (32 bit)	PTC - **ObjectAda**
MS Windows NT+	Intel x86 (32 bit)	SofCheck - **AdaMagic**
MS Windows NT+	Intel x64 (64 bit)	
Linux	Intel x86 (32 bit)	
Mac OS X	PowerPC (64 bit)	
Mac OS X	Intel x64 (64 bit)	
Solaris	SPARC (32 or 64 bit)	
Solaris	Intel x64 (64 bit)	

Overview - milestones (full list here)

From day one Zip-Ada is a free, open-source project.



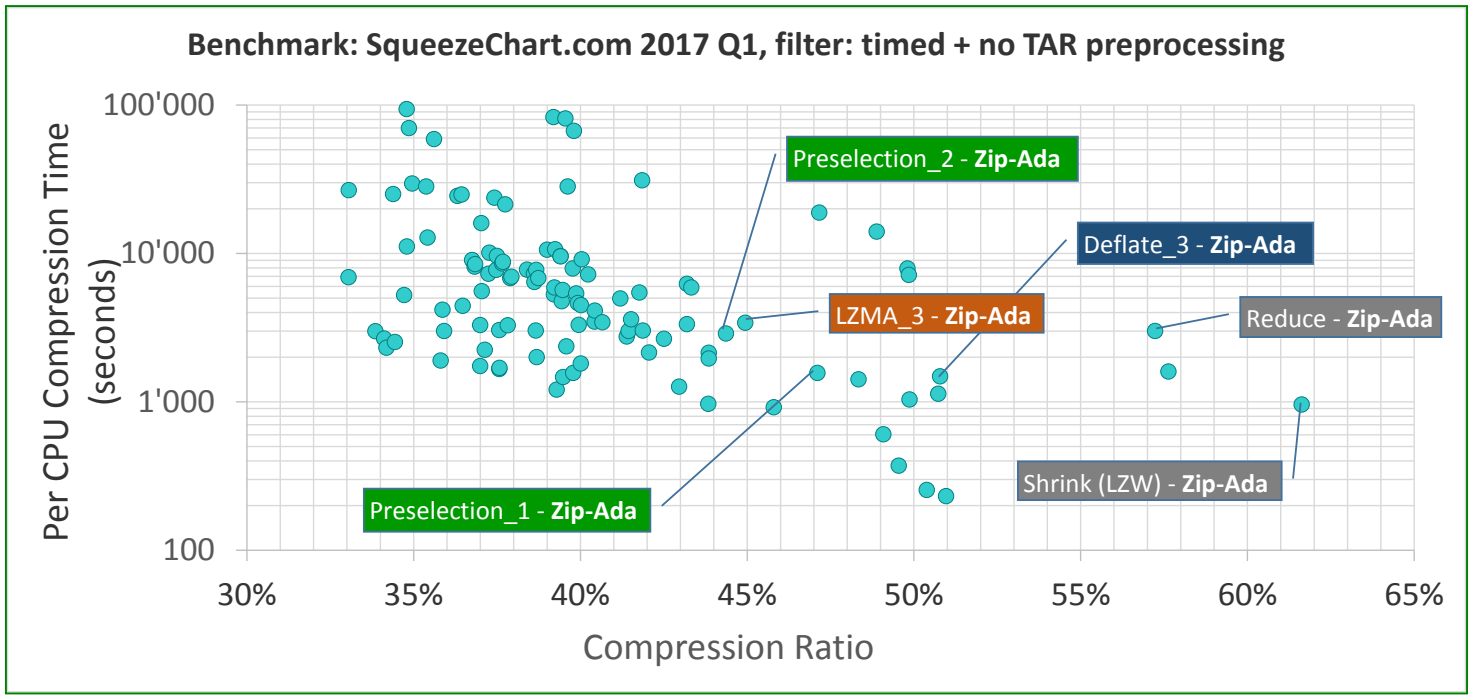
- **1999**: started with **decompression** only
- **2007**: SourceForge hosting, **http://unzip-ada.sf.net/**
- **2007**: added **1st compression** method (LZW) from a Pascal source
- **2008**: full streams support (**contrib.** NXP semiconductors)
- **2009**: added BZip2 decompression (*)
- **2010**: profiling, stream performance, UTF-8 (**contrib.** Romans CAD)
- **2011**: developed a simple Deflate method for compression
- **2014**: added LZMA decompression (*) from reference decoder
- **2016**: developed an advanced **Deflate** compression method
- **2016**: developed a **LZMA** compression method (*)

(*) can be used standalone

What is a “good” compression in general ?

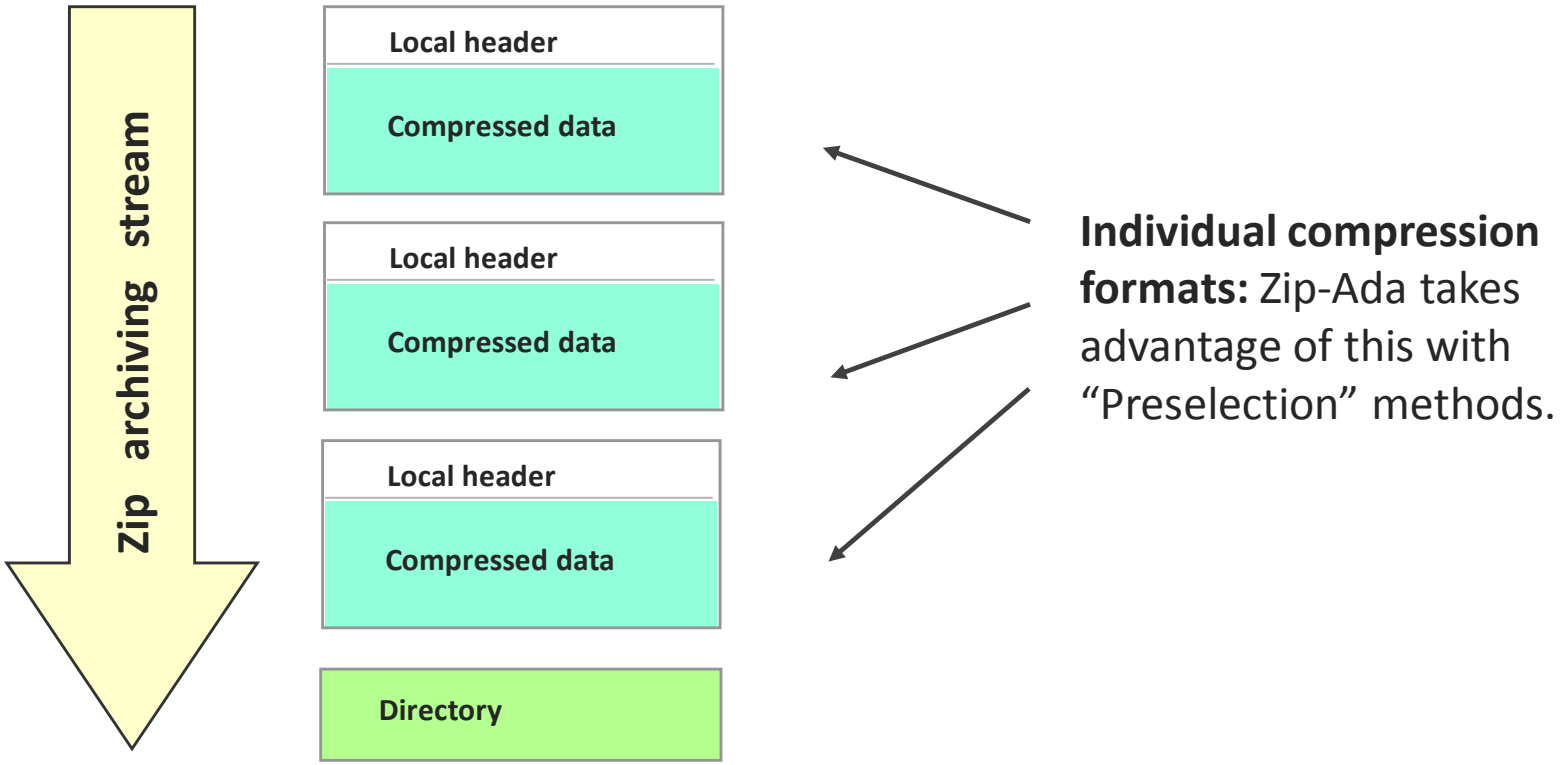
No easy answer: there are multiple criteria!

1. Compression ratio
2. Per-CPU decompression time (dep.: format and compressed size)
3. Per-CPU compression time (dep.: format symmetry, algorithm, effort)
4. Memory footprint



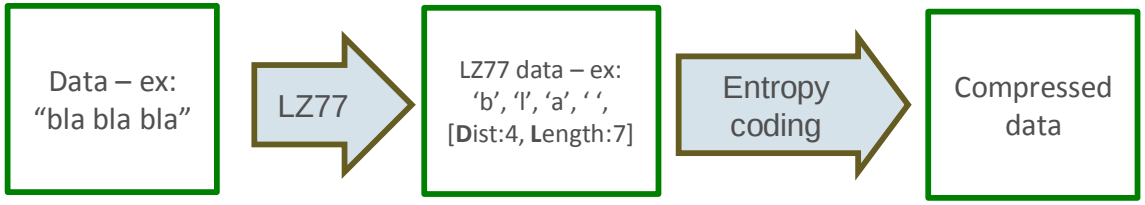
The Zip archive format

- Origin: Phil Katz's PKZIP (~ 1989) – old, limited… but used everywhere.
- **Multi-file** data archive container format with compression.
- **Open** regarding compression formats: Store, LZW, Deflate, LZMA, …



The Deflate format

- Invented by Phil Katz (1989)) – old… but used everywhere.
- Combines **LZ77** (font-end) and **Huffman trees** (entropy back-end).



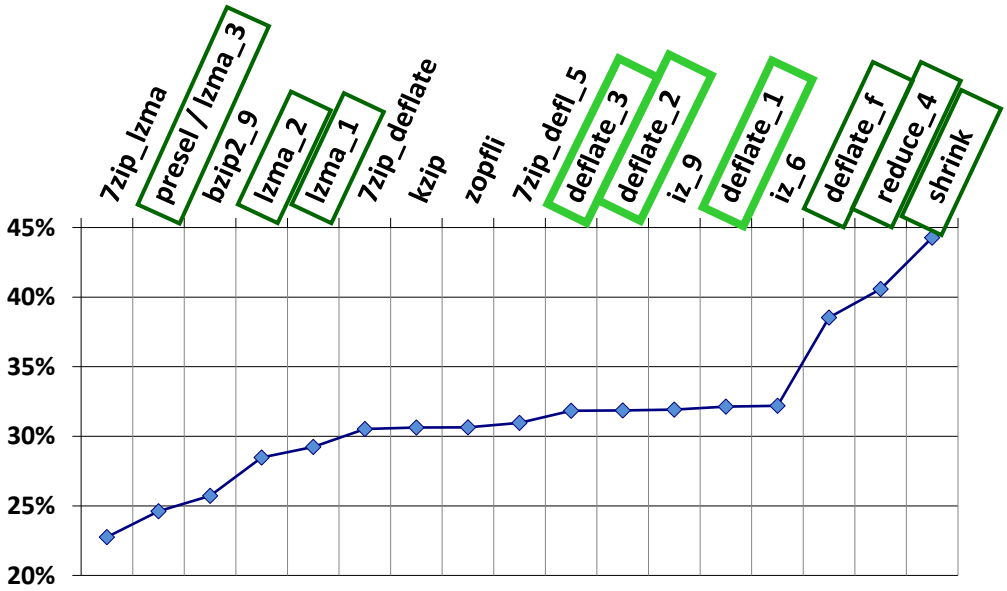
- **Multi-block** (choice of blocks can be adapted to contents); a block is either:
  1. LZ77 data sent with **predefined Huffman trees**; end-of-block symbol
  2. or LZ77 data sent with **ad-hoc Huffman trees**, preceded by a **header** describing the trees (tradeoff: the use of adapted trees saves room; but each header takes some room); end-of-block symbol
  3. or a fallback solution: data just stored, when it is too random.
- Within a block: **not adaptive**: same trees used until block termination.

The Deflate format – Zip-Ada implementation – 2016

- Common generic **LZ77** (can be used standalone), including Info-Zip/zlib implementation – this one is used for our Deflate.
- Optimal **Huffman** tree construction (generic, standalone as well) taken from Zopfli (ref. #3).
- Genuine block-cutting algorithm ("Taillaule") using similarities of bit length vectors – that's the "smart" thing that we have developed here.

Silesia corpus			
Date / Size	% compr	Name	Deflate bench
48'240'494	22.8%	7zip_lzma	-25.44%
52'169'187	24.6%	presel / lzma_3	-19.37%
54'509'539	25.7%	bzip2_9	-15.75%
60'346'016	28.5%	lzma_2	-6.73%
61'970'916	29.2%	lzma_1	-4.22%
64'698'142	30.5%	**7zip_deflate**	**0.00%**
64'921'533	30.6%	kzip	+0.35%
64'949'384	30.6%	zopfli	+0.39%
65'636'076	31.0%	7zip_defl_5	+1.45%
67'462'614	31.8%	deflate_3	+4.27%
67'506'579	31.9%	deflate_2	+4.34%
67'634'472	31.9%	iz_9	+4.54%
68'110'939	32.1%	deflate_1	+5.27%
68'230'447	32.2%	iz_6	+5.46%
81'667'070	38.5%	deflate_f	+26.23%
85'991'264	40.6%	reduce_4	+32.91%
93'826'501	44.3%	shrink	+45.02%
211'938'580	100.0%	original data	

Green = Zip-Ada



The Deflate format – Taillaule algorithm

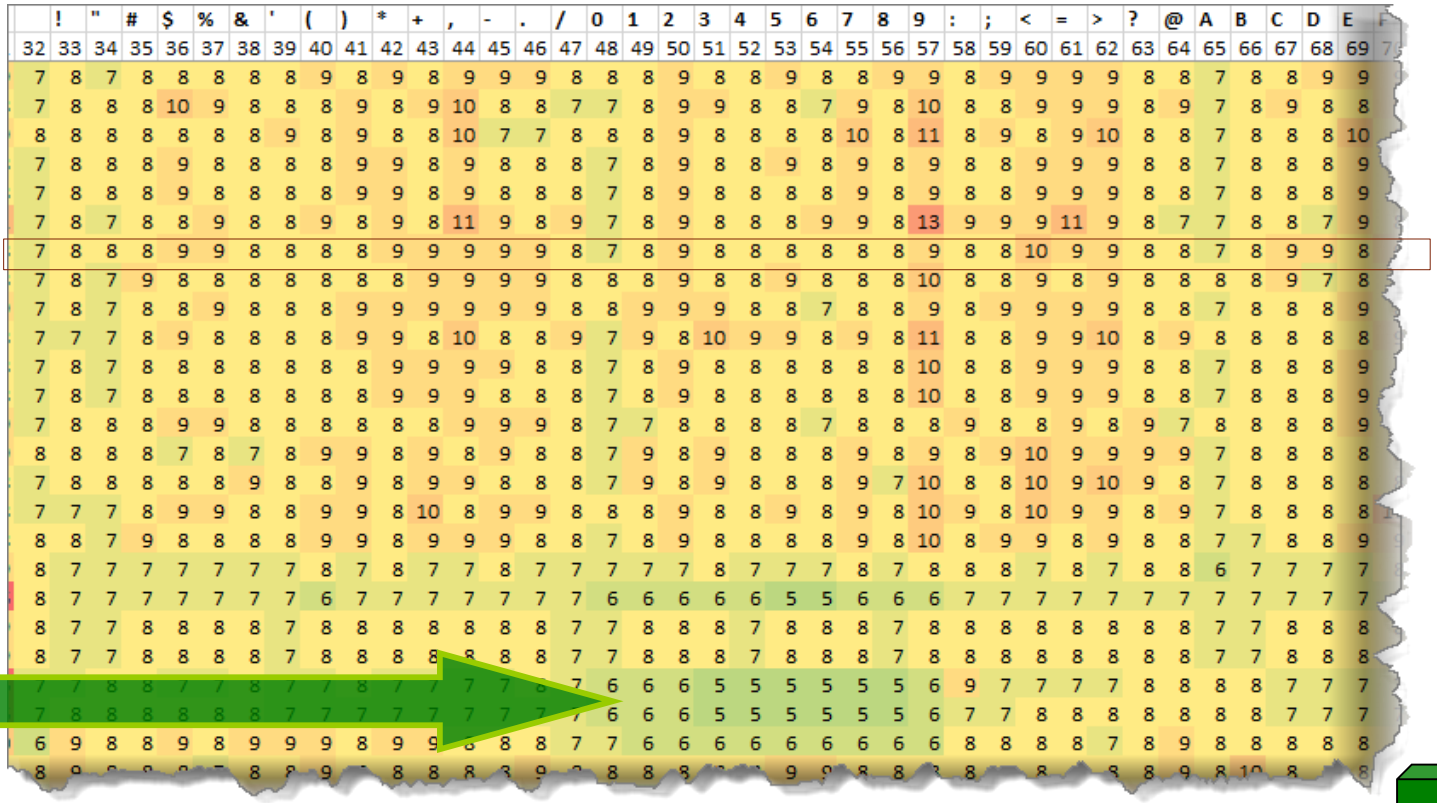
Huffman trees are uniquely determined by the set of *bit lengths* (the travel from root to leaf for each symbol). Consequently, only *bit lengths* are stored as compression structures. Our single-pass algorithm detects changes in the data stream “on the fly” by comparing *bit length* vectors. L1 norm seems the best, so far.

LZ77 symbols

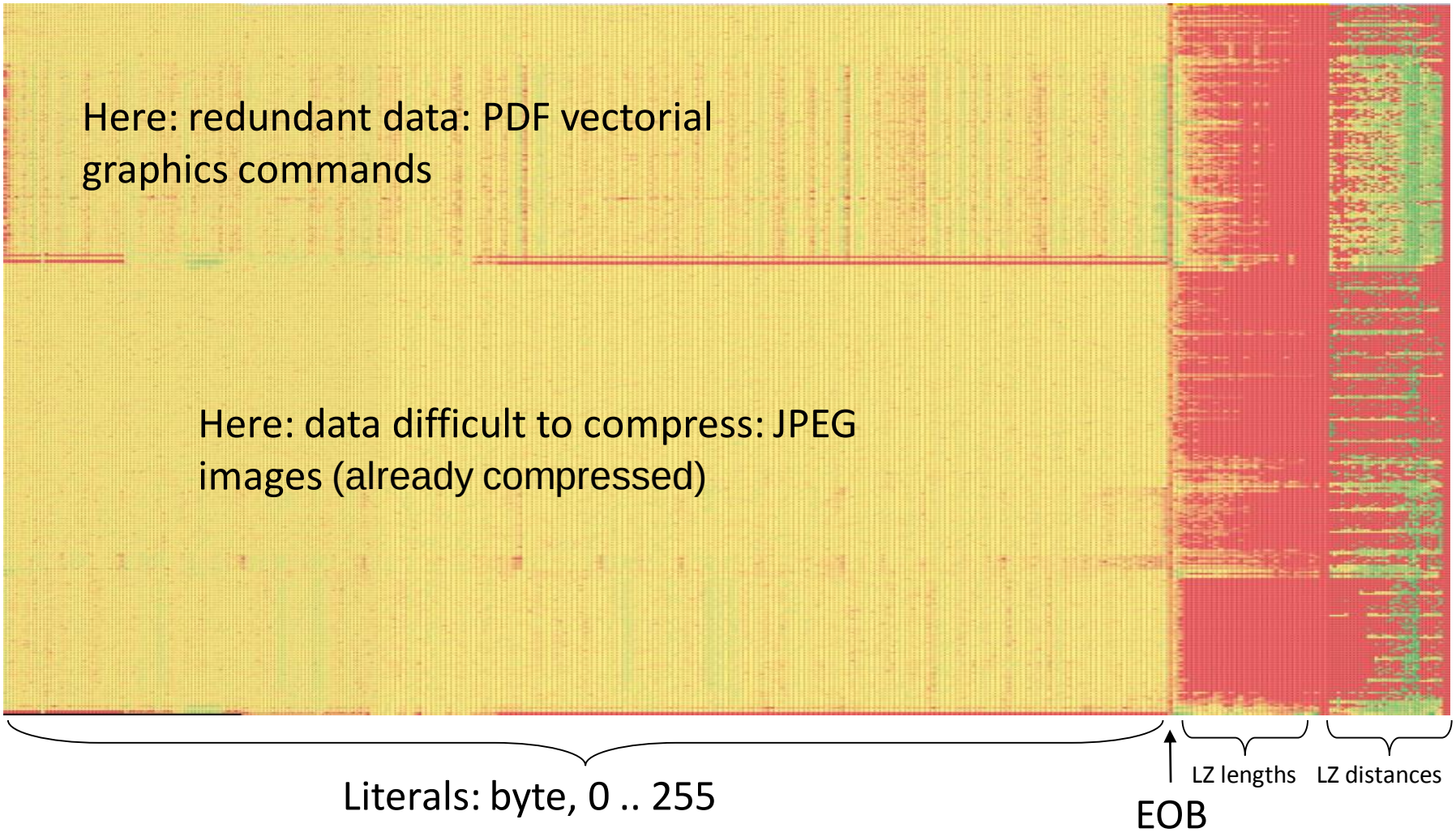
Optimal bit lengths based on statistics of chunks of LZ77 data.

1 row = 1 vector of bit lengths for a data chunk

Frequent symbol
⇒ Low bit length
⇒ Better compression



Annex – Example of bit lengths for a PDF file



References

1. **Zip-Ada** web site **http://unzip-ada.sf.net/**

2. **AZip** web site **http://azip.sf.net/** (AZip is GUI archive manager using Zip-Ada)

3. **Squeeze Chart:** large and varied corpus: 5 GB; 21,532 files; web site: **http://www.squeezechart.com/**

4. **A fast and space-economical algorithm for length-limited coding**, Katajainen J., Moffat A., Turpin A. (1995), Lecture Notes in Computer Science, vol 1004. Springer, Berlin, Heidelberg

5. **DEFLATE Compressed Data Format Specification version 1.3**, P. Deutsch, 1996, **https://www.ietf.org/rfc/rfc1951.txt**

6. **Zip file format specification:** **https://support.pkware.com/display/PKZIP/APPNOTE**