Zip-Ada

bzip2

Advent of Compression

Writing a working BZip2 encoder in Ada from scratch in a few days

https://alire.ada.dev/crates/zipada

Dr Gautier de Montmollin

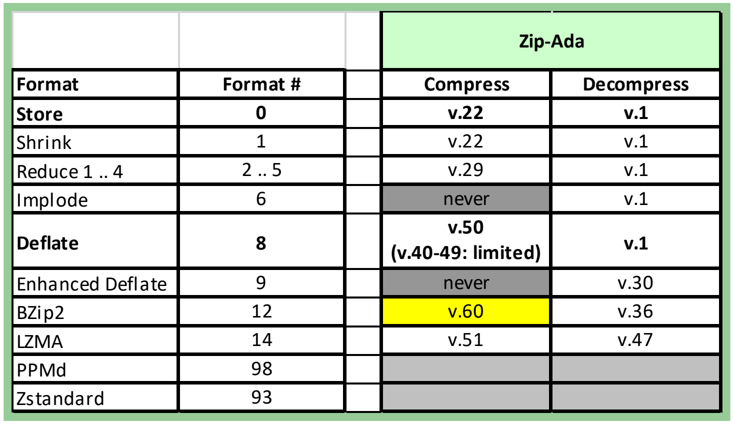
FOSDEM 2025

Zip-Ada

bzip2

Motivations:

- fun / challenge / warm-up for Advent of Code 2024
- fill a gap in the Zip-Ada compatibility grid:



			Zip-Ada	
Format	**Format #**		**Compress**	**Decompress**
Store	**0**		**v.22**	**v.1**
Shrink	1		v.22	v.1
Reduce 1 .. 4	2 .. 5		v.29	v.1
Implode	6		never	v.1
Deflate	**8**		**v.50** **(v.40-49: limited)**	**v.1**
Enhanced Deflate	9		never	v.30
BZip2	12		v.60	v.36
LZMA	14		v.51	v.47
PPMd	98			
Zstandard	93			

Expectations (low):

- BZip2 compresses few kinds of files better than, for instance, LZMA
- BZip2 compression scheme is mostly “mechanical”: on most steps, there is only one single possible encoding.
- BZip2 is a weakened version of BZip1 (old patent issues)

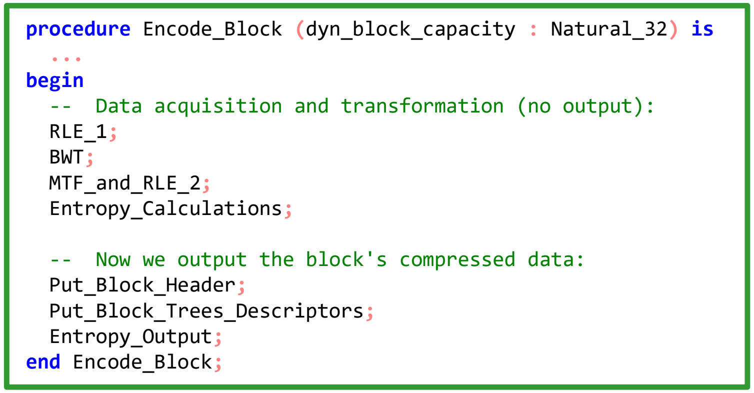
Results: two very good surprises!

Zip-Ada

bzip2

BZip2 is very simple.

1. Input: a “large” block of data (<= 900 KB)
2. The block is processed “off-line”
   - Run Length Encoding (2x)
   - Burrows-Wheeler Transform (**b**lock-sorting)
   - Move To Front
   - Entropy coding (Huffman)
3. Output of the compressed block.



```
procedure Encode_Block (dyn_block_capacity : Natural_32) is
  ...
begin
  --  Data acquisition and transformation (no output):
  RLE_1;
  BWT;
  MTF_and_RLE_2;
  Entropy_Calculations;

  --  Now we output the block's compressed data:
  Put_Block_Header;
  Put_Block_Trees_Descriptors;
  Entropy_Output;
end Encode_Block;
```

Run Length Encoding #1

a	⟶	a
aa	⟶	aa
aaa	⟶	aaa
aaaa	⟶	aaaa[0]
aaaaa	⟶	aaaa[1]
aaaaaa	⟶	aaaa[2]
…		

1	⟶	1
2	⟶	2
3	⟶	3
4	⟶	5
5	⟶	5
6	⟶	5
…	⟶	5
259	⟶	5

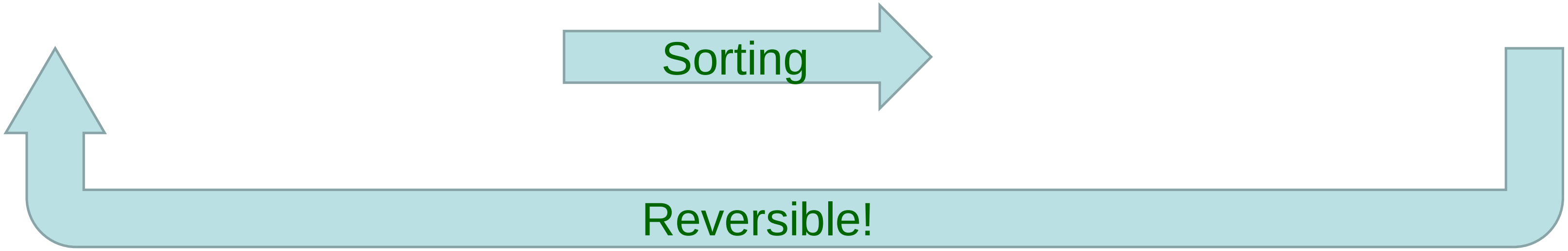
Burrows-Wheeler Transform

```
Mary had a little lamb, its fleece was white as snow
ary had a little lamb, its fleece was white as snowM
ry had a little lamb, its fleece was white as snowMa
y had a little lamb, its fleece was white as snowMar
 had a little lamb, its fleece was white as snowMary
had a little lamb, its fleece was white as snowMary 
ad a little lamb, its fleece was white as snowMary h
d a little lamb, its fleece was white as snowMary ha
 a little lamb, its fleece was white as snowMary had

…
```

```
a little lamb, its fleece was white as snowMary had
as snowMary had a little lamb, its fleece was white
fleece was white as snowMary had a little lamb, its
had a little lamb, its fleece was white as snowMary
its fleece was white as snowMary had a little lamb,
lamb, its fleece was white as snowMary had a little
little lamb, its fleece was white as snowMary had a
snowMary had a little lamb, its fleece was white as
was white as snowMary had a little lamb, its fleece

…
```

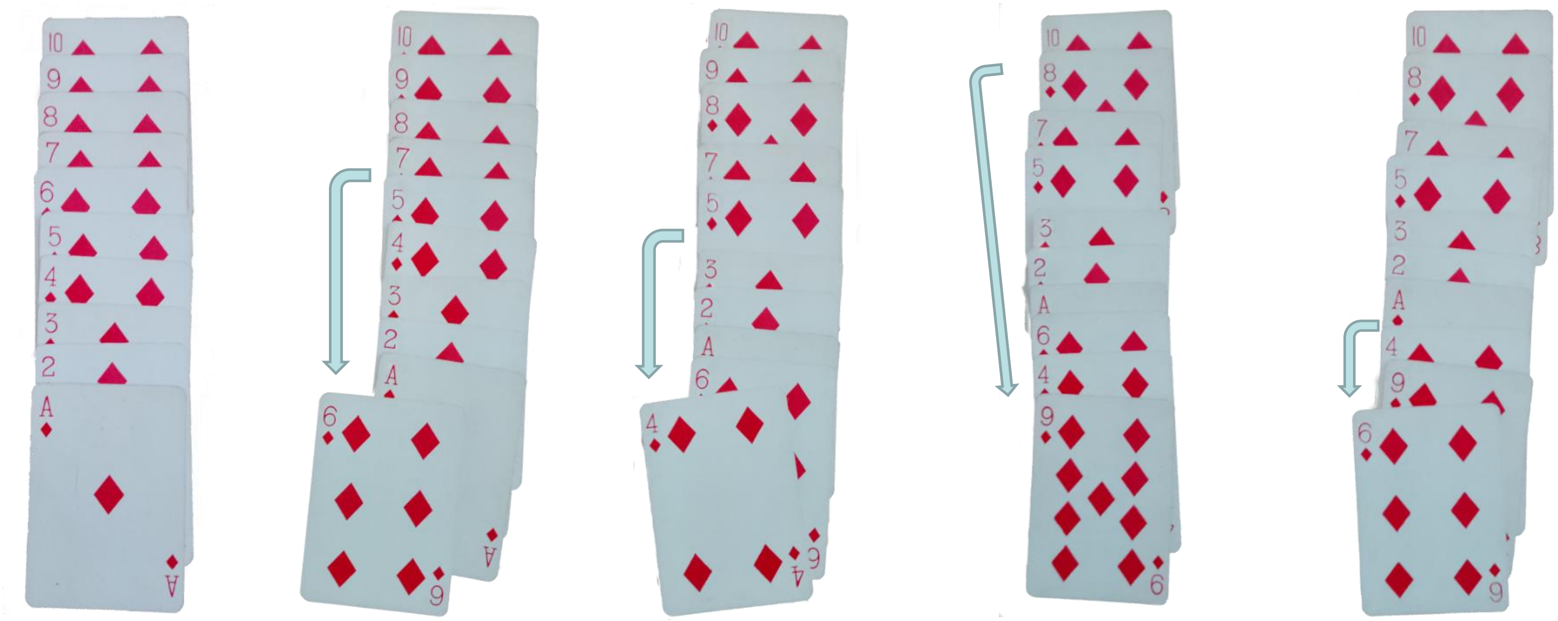


Burrows-Wheeler Transform (continued)

Output of bzip2-encoding.adb (excerpt):

```
2)_____FPUAAEOOA(     E  _______'_'''''_'''''''''__________________________   ___( 8)  ____  __
________RSGAESREIIOUI I   EE ( 6)  ( 4)   ( ______       ( 8)( 10)  ___   ( 6)__ ______    "  _______-
_______( 8)( 8)( 4)( 10) ( 8)        ( 4)( 4)( 8)( 8)( 8)______  _   _  ..
HOIUURNUWWCWNWWWWNIMMMMMMMMMMMMMI      OOEOUIRNCCI  NFFFNIII___________   ...         ___( 6)( 10)(
12)( 8)( 8)( 8)( 8)( 4)( 2)( 4)( 6)( 6) ( 8)     ___     ____________LCRRRRRRP BOOA.. ...( 4)( 0)( 4)
(((( ( (   (._______ _________EO''_ __   __ BTTT   R  SBBBBBBB      -( 2)  ( 0)  ( 0)ENNTTTPT___ -(
0)BBBBBBBBBBBBB-
'11EkkkeeeeeeeeeEEE22E0EEEEdrrddldrrllddlllrdllllllllleeeeeleellllreledlllrlddlrellled0kkkdkkkkNNNeryer
erryxxxxxxtFeettthhteFgtgthFtghtFFdddrrNNNhhhhnnnnnnttttttttttttttttttttttttttttttttteeeetttttttttttedee
eyyywwwwlllllllllllllyyyyyyyyyyyyyyyyyyyyyyeddcccslsslsssldddeeeesss22eeeeeetmmmdddddtttooeeeeeeeeekmmk
mkttteeetttrrttennntttttthhtw22ggeeetttyyyyggggggnyyyttdddFFrrrrtrtttrrrrtrtttrrrrtrrrrryyyyyyygggdddte
eeeeeeeeettteeeeeemmgggyyyyytgtgyrrrrrreeeeeeeodgttttgtgttteeekkketxyyxsettsennnnnnnnnnnnnnnnhhheedddnn
nxFFFeeenttttwwttttttwwttttwwwwtttxnnnnnlllllbtttttgggggggrrrnnkkkkkkkknhnhtttqqnttnrqtqntrrnnnnrrttff
fffffffffffffxxxxxxxxxxxxxxxxxxeeyyyyyyyyyyyyyyyyyyyyyyyyytttccrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrcrrsssssss
sssssssssssssssssssssssssskkkkdkkddkkkddkkktfffffffffffftftftttrrrrrrrtttxxmplllplpppleekmeeflllllfrffl
lllllfrfrllllllfflffffffffrfrffrlllllfflllllffffffflllrlrttttttttttttttttttlllllsslssslllsslltttttnnffffff
ffffffffffeeeeeeeeeeeeeeeeeeetttttttttttttttttttttttttttttttttttttttyyyyyynnrrryypppptttgggppeeeepppppptt
tttttbbbbbkkkkmmmmmmmttttllttttootottteekkkktkkkkpknpppkknkpkkkkpppppppppppppeeeffffyyffyffttttttttllll
lllcnhhnchceeqqnnnnnnnnnnnnnnnnnnnnnnnnnnnnrrrrrrrrreeeeeeeeeeeeeee
```

Move To Front



Card: 6
Index: 6

Card: 4
Index: 5

Card: 9
Index: 9

Card: **6**
Index: **3**

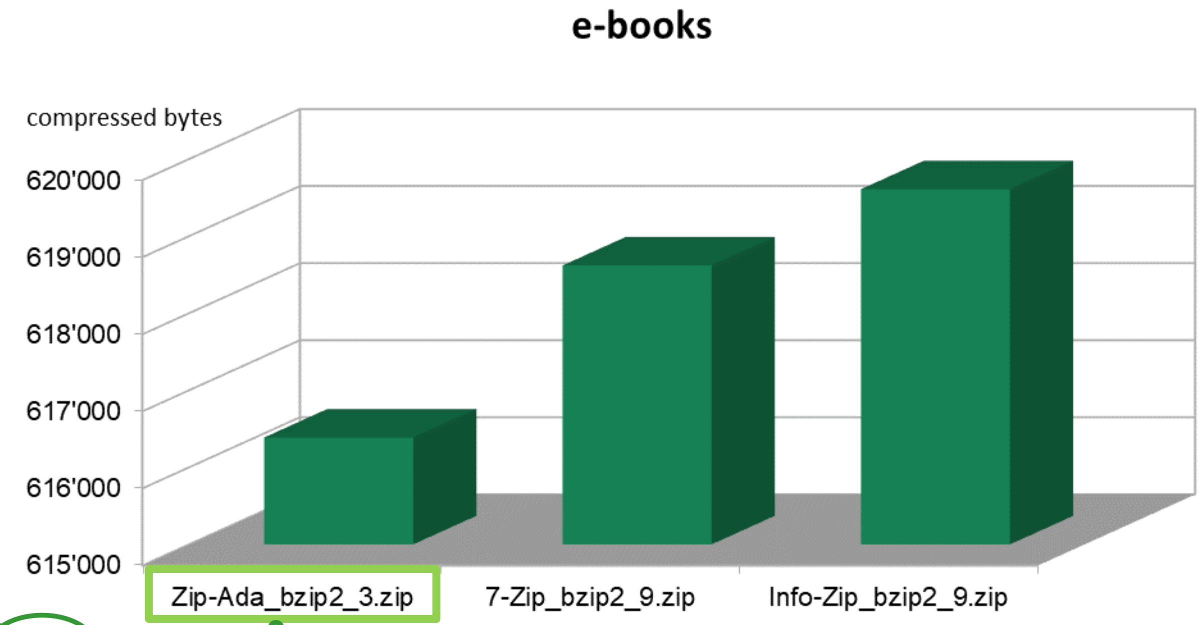
Final step: entropy coding with Huffman trees

Not mechanical. You have up to 6 trees, *freely* defined, that can be *freely* chosen for each group of 50 symbols (the output of Move To Front)

⟶ Room for **optimization**!

Results – first surprise

Zip archive, BZip2 only:



Zip-Ada
wins bigly !!!

NB: BZip2 is very good with (at least) human-written **texts** and **source code**.

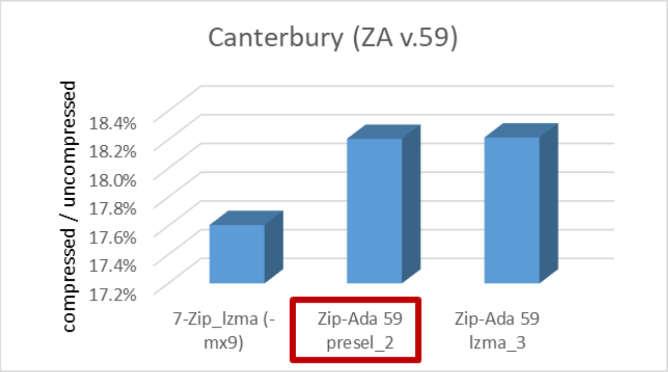
Zip-Ada

bzip2

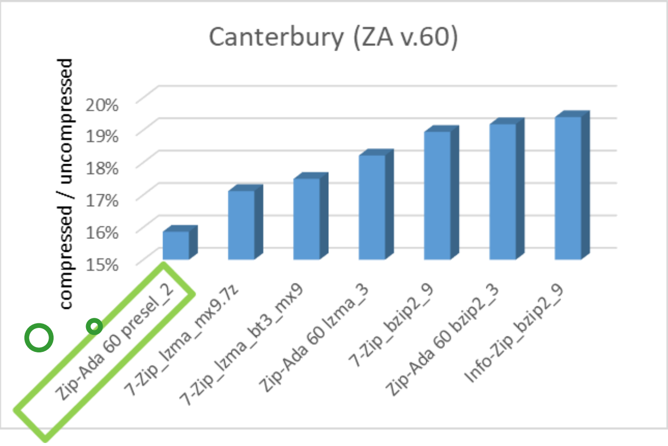
Results – second surprise

Zip archive, multi-format (for Zip-Ada, Preselection_2):

Before:



After:



Zip-Ada wins here too !!!

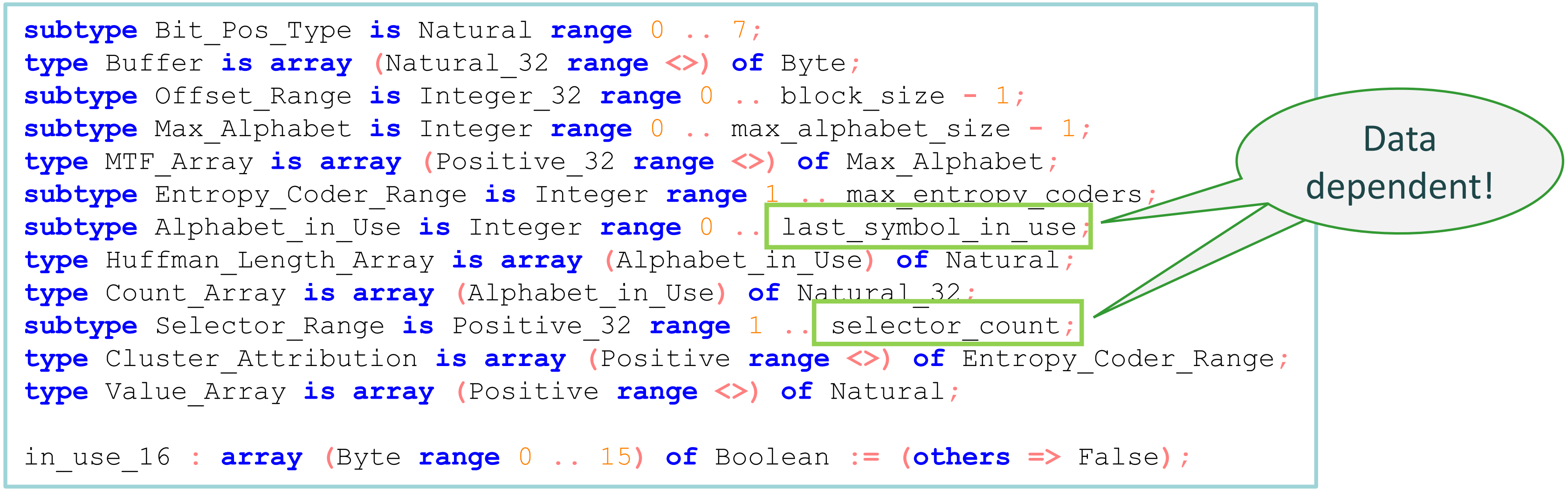
Benefits of Ada

Data compression is very difficult to debug, sometimes impossible.

⟶ Ada does its best to help you doing things right the first time.

Indirect benefit : you can focus on the algorithms.

Here, some **ranges** picked up from the code (bzip2-encoding.adb):



```
subtype Bit_Pos_Type is Natural range 0 .. 7;
type Buffer is array (Natural_32 range <>) of Byte;
subtype Offset_Range is Integer_32 range 0 .. block_size - 1;
subtype Max_Alphabet is Integer range 0 .. max_alphabet_size - 1;
type MTF_Array is array (Positive_32 range <>) of Max_Alphabet;
subtype Entropy_Coder_Range is Integer range 1 .. max_entropy_coders;
subtype Alphabet_in_Use is Integer range 0 .. last_symbol_in_use;
type Huffman_Length_Array is array (Alphabet_in_Use) of Natural;
type Count_Array is array (Alphabet_in_Use) of Natural_32;
subtype Selector_Range is Positive_32 range 1 .. selector_count;
type Cluster_Attribution is array (Positive range <>) of Entropy_Coder_Range;
type Value_Array is array (Positive range <>) of Natural;

in_use_16 : array (Byte range 0 .. 15) of Boolean := (others => False);
```

Data dependent!

More @ https://gautiersblog.blogspot.com/2024/11/writing-bzip2-encoder-in-ada-from.html