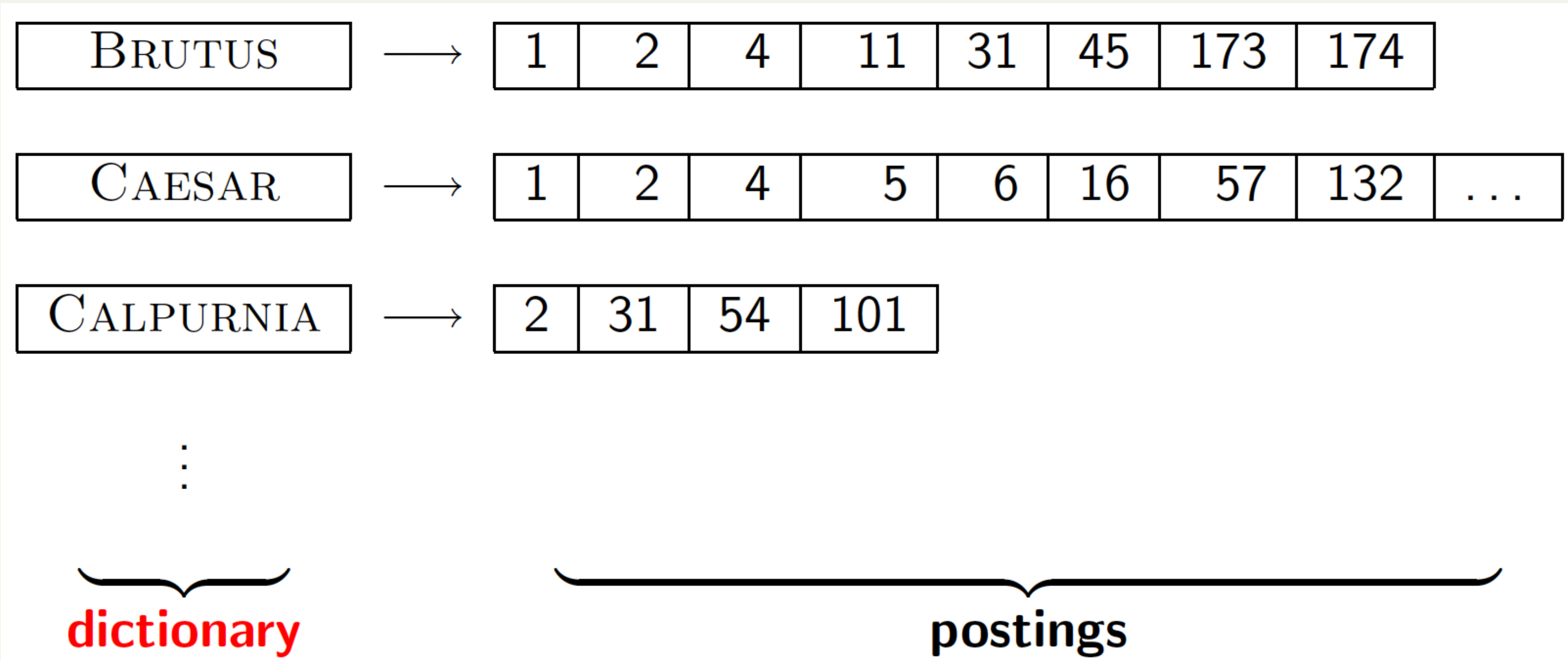
Index construction: Compression of postings

Paolo Ferragina
Dipartimento di Informatica
Università di Pisa

Gap encoding

1 1 2 7 20 14 …



Then you compress the resulting integers with variable-length prefix-free codes, as follows…

Variable-byte codes

- Wish to get very fast (de)compress → byte-align
- Given a binary representation of an integer
  - Append 0s to front, to get a multiple-of-7 number of bits
  - Form groups of 7-bits each
  - Append to the last group the bit 0, and to the other groups the bit 1 (tagging)

e.g., $v=2^{14}+1$ → *binary(v)* = 100000000000001

1*0000000*1 10000000 00000001

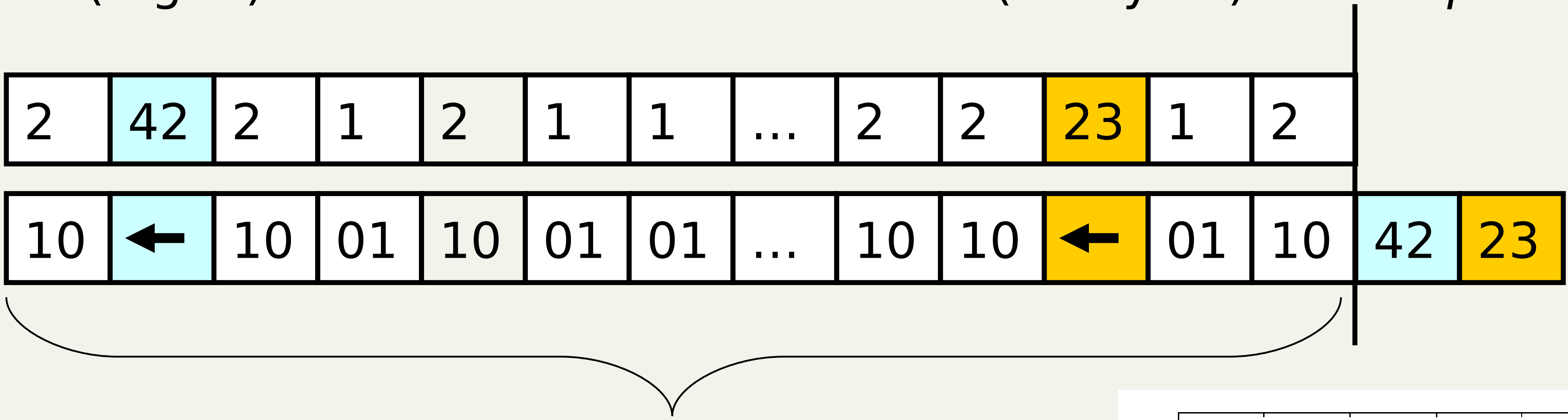
Note: We waste 1 bit per byte, and avg 4 for the first byte.

But it is a prefix code, and encodes also the value 0 !!

T-nibble: We could design this code over t-bits, not just t=8

PForDelta coding

Use b (e.g. 2) bits to encode 128 numbers (32 bytes) or *exceptions*



a block of 128 numbers = 256 bits = 32 bytes

Translate data: [base, base + 2^b**-2**] → [0,2^b **- 2**]

Encode exceptions with value **2^b-1**

Choose b to encode 90% values, or trade-off:

b↑ waste more bits, b↓ more exceptions

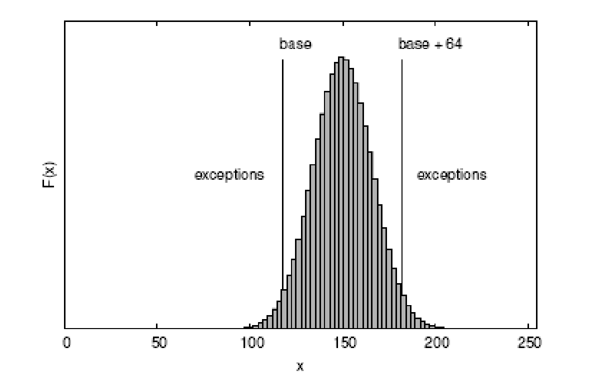


Figure 4.13: PFOR compression for $b = 6$ and $base = 118$ captures most of the values of this 256 value domain as codes.

γ-code

0000...........0	x in binary
Binary Length-1	Binary length

- $x > 0$ and Binary length $= \lfloor \log_2 x \rfloor + 1$

e.g., 9 represented as 0001001.

- γ-code for x takes $2 \lfloor \log_2 x \rfloor + 1$ bits

(ie. factor of 2 from binary)

It is a prefix-free encoding...

- Given the following sequence of γ-coded integers, reconstruct the original sequence:

0001000001100110000011101100111

8 6 3 59 7

Elias-Fano

1 =	000	01
4 =	001	00
7 =	001	11
18 =	100	10
24 =	110	00
26 =	110	10
30 =	111	10
31 =	111	11

z = 3, w=2

B = 010010010000000000100000010100011

x[0…7] = {1,4,7,18,24,26,30,31}

Represent numbers in ceil[log m] bits, where m= |B|

Set z = ceil[log n] and where n = #1 then it can be proved

- *L takes ≅ n log (m/n) bits*
- *H takes = n 1s + n 0s = 2n bits*

L = 0100111000101011

0	1	2	3	4	5	6	7

In unary H = 10|110|0|0|10|0|110|11

How to get the i-th number ? Take the i-th group of w bits in L and then represent the value ((*pos of i-th 1*) – i) in z bits