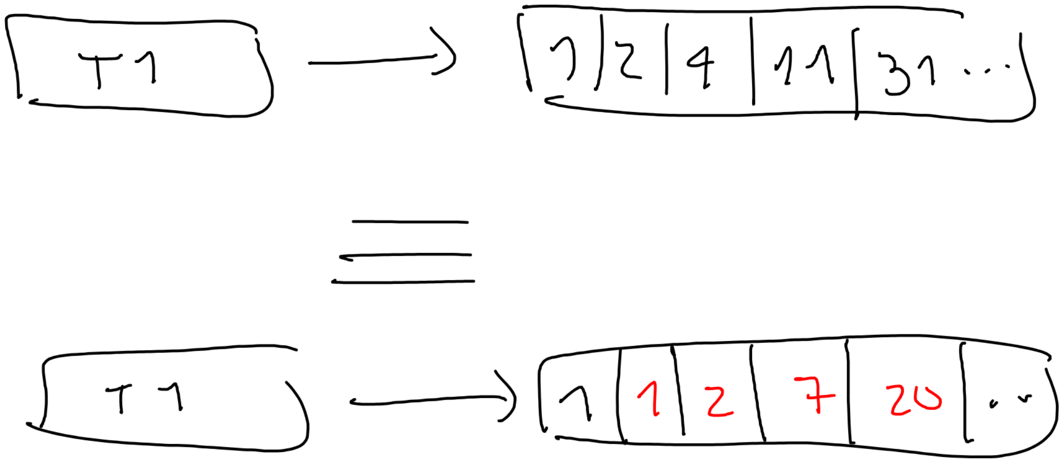
Posting List compression

Since a posting list is a crescent list of docIDs, which are integers, we can use several coding techniques to reduce the amount of used space

GAP ENCODING

Compress the docID by storing only the integer distance with the root of the list



Useful for the other codings too!

GAMMA CODING (γ)

→ Given an integer value (in our case, a docID), compute its binary reppresentation, that will require n bits

→ The coded integer will be

$$\underbrace{0\ldots\ldots0}_{n-1 \text{ zeroes}} \; ; \; \text{binary reppresentation}$$

EXAMPLE

docID = 4

gamma code = $00 \,|\, 100$

Delta Code (δ)

→ Given an integer value (in our case, a docID), compute its binary representation, that will require n bits

→ The coded integer will be

Gamma encoding of the length of the binary representation | binary representation

EXAMPLE

docId = 4

$$\underbrace{011}_{\text{Gamma code }(3)\ =\ 0|11}|100$$

P for Delta

→ Given array of integer, can be unordered

→ Pick a base and n (number of bits)

→ Compute the difference between the base and each element

→ Represent values in the range $[0, 2^n - 1]$ with their binary representation, represent everything else in "excess" with the "Escape" sequence 2^n

$S = \{2, 5, 10, 1, 3\}$ base $= 1$, $n = 2$

$S - \text{base} = \{1, 4, 9, 0, 2\}$

Value to code in range $[0, 2^2 - 1]$:

$\text{coded\_}S = 01 \vdots 11 \vdots 11 \vdots 00 \vdots 10$

excess

Elias Fano

ENCODE

→ Given ordered integer array (crescente)

→ Take U, the power of two bigger than the array max value (the last one)

→ Calculate $w = \lceil \log_2 \frac{U}{n} \rceil =$

$= \lceil \log_2 U - \log_2 n \rceil$

→ Calculate $z = \lceil \log_2 U \rceil - w$

→ Write the binary representation of the elements using $\underbrace{\log_2 U}$ bits (add padding in case) → number of bits of U

→ Partition the representation in z bit and w bit, starting from right

→ Compute the following bit string:

L = concat of all bit in range w

$H = 2^z$ buckets, each separated by a 0 (sort of '\0') and filled with as many 1 as the number of element falling in the buckets in range z

Obs: If the sequence in input is not ordered, we "complementary" gap encoding

$$\{1,1,3,4,2\} \longrightarrow \{1,2,5,9,11\}$$

DECODE

We want to get the k-th integer

→ Position L in bucket of length w and take the k-th bucket

→ Define Select(p, H) = position of p-th bit 1 in H (ignoring 0s!!)

→ Select$(k, H) - k$ and transform it in binary

→ Concat the prev value with the k-th bucket of L

→ Convert in integer