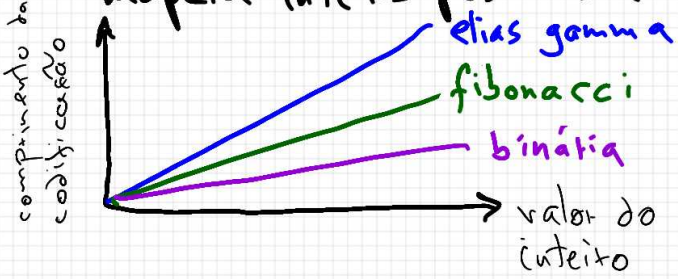
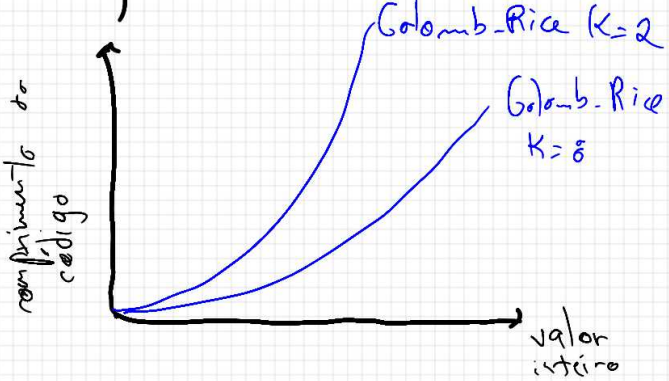
Codificação

- códigos pré-fixados
  um codeword qualquer NÃO é prefixo de algum outro codeword
  dispensa delimitadores
- codificação universal (para inteiros)
  mapeia inteiros positivos em codewords binários



para qualquer valor inteiro, o codeword sempre vai estar a um fator constante do comprimento ótimo

Codificações não-universais: Golomb, Golomb-Rice, unária ...



onde K é o valor do divisor

Golomb 66 Run-Length encodings: Transaction on Info. Theory

UNÁRIO BINÁRIO Supondo divisor $K = 4$

símbolo (int) → CODEWORD: PREFIXO | STOP BIT | SUFIXO

$\lg 4 = 2$ → tamanho do resto (em bits)

	PREFIXO	STOP BIT	SUFIXO
0		1	00
1		1	01
2		1	10
3		1	11
4	0	1	00
5	0	1	01
6	0	1	10
7	0	1	11
8	00	1	00
9	00	1	01
10	00	1	10
11	00	1	11
⋮			

Golomb-Rice: divisor K é potência de 2.

Usado em (JPEG-LS, FLAC) em conjunto com MTF e BWT

Golomb (continuação)

Decoder deve conhecer o valor de K (divisor), o encoder pode também transmitir esta informação para o decoder...

Decodificação

① lê o prefixo (a quantidade de zeros até o stop bit) → conta quantos zeros

② lê o sufixo em binário
sabe quantos bits devem ser lidos porque sabe qual é o valor do divisor K → $\lg K$ = tamanho do resto

→ +

Elias-Gamma:

2^N + resto em binário

onde N é o maior potência de 2 dentro do número a ser codificado

$int(\lg num) = N$

PREFIXO: N em unário

SUFIXO: resto em binário

		STOP BIT
1	$2^0 + 0$	1
2	$2^1 + 0$	0 1 0
3	$2^1 + 1$	0 1 1
4	$2^2 + 0$	0 0 1 0 0
5	$2^2 + 1$	0 0 1 0 1
6	$2^2 + 2$	0 0 1 1 0
7	$2^2 + 3$	0 0 1 1 1
8	$2^3 + 0$	0 0 0 1 0 0 0
⋮	⋮	⋮

DECODE:

① lê nºs de zeros (prefixo) → = N

② lê bits do sufixo (resto) e soma ao N

Obs: não codifica diretamente o ZERO!

Ex: bitmagic C++ lib

Codificação Fibonacci

Teorema de Zeckendorf

Qualquer inteiro pode ser representado como a soma de valores de termos Fibonacci (não consecutivos)

A série:

-1 1 0 1 1 2 3 5 8 13 21 34 ...

Ex: 40 ?

1 0 0 1 0 0 0 1 1

11 ?

0 0 1 0 1 1

STOP BIT: não há 1's consecutivos na codificação

Obs.: robusto contra erros de inserção/remoção

→ self-synchronizing

Huffman

Ex.: a b a b b e b c a a e a a b a d d a e c c c b e a a a a e e e e d e c a e a a

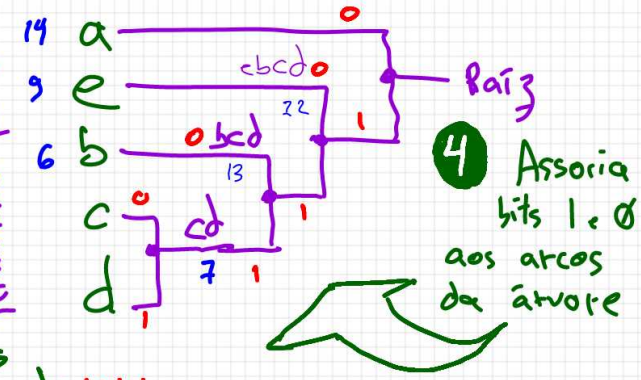
1. contabilização:
   - a: 14
   - b: 6
   - c: 4
   - d: 3
   - e: 9

2. ordena (decrescente) símbolos por frequência:

14	a
9	e
6	b
4	c
3	d

símbolos folhas / terminais

3. monta árvore bottom-up a partir dos símbolos de menor frequência



4. Associa bits 1 e 0 aos arcos da árvore

5. codewords
   - a: 0
   - b: 110
   - c: 1110
   - d: 1111
   - e: 10