

Licenciatura em Engenharia Informática

# Sistemas Multimédia

## Codificação de Dados sem Perdas

Telmo Reis Cunha

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro – 2020/2021

# 1. Codificação de Dados

---

- No módulo anterior foi analisada a codificação ao nível do símbolo, através de códigos binários.
- Agora será analisada a codificação de dados (guardados em ficheiro, ou constituindo mensagens a ser transmitidas).
- Por exemplo, pode-se tirar partido de, num texto, várias palavras (sequências de símbolos) aparecerem repetidas várias vezes, codificando o texto de uma forma mais compacta (a repetição de tais palavras é, assim, codificada de uma forma mais eficiente).
- Consideram-se, apenas, esquemas de codificação de dados sem perda de informação.

## 2. Codificação RLE (Run-Length Encoding)

---

- Esta técnica de codificação consiste em representar cada ocorrência de um símbolo seguido do número de vezes que ele surge repetido nessa posição.

- Por exemplo, a sequência de símbolos:

AAAAAAAAABBBBCCCCCCCCCCCCBBABBB

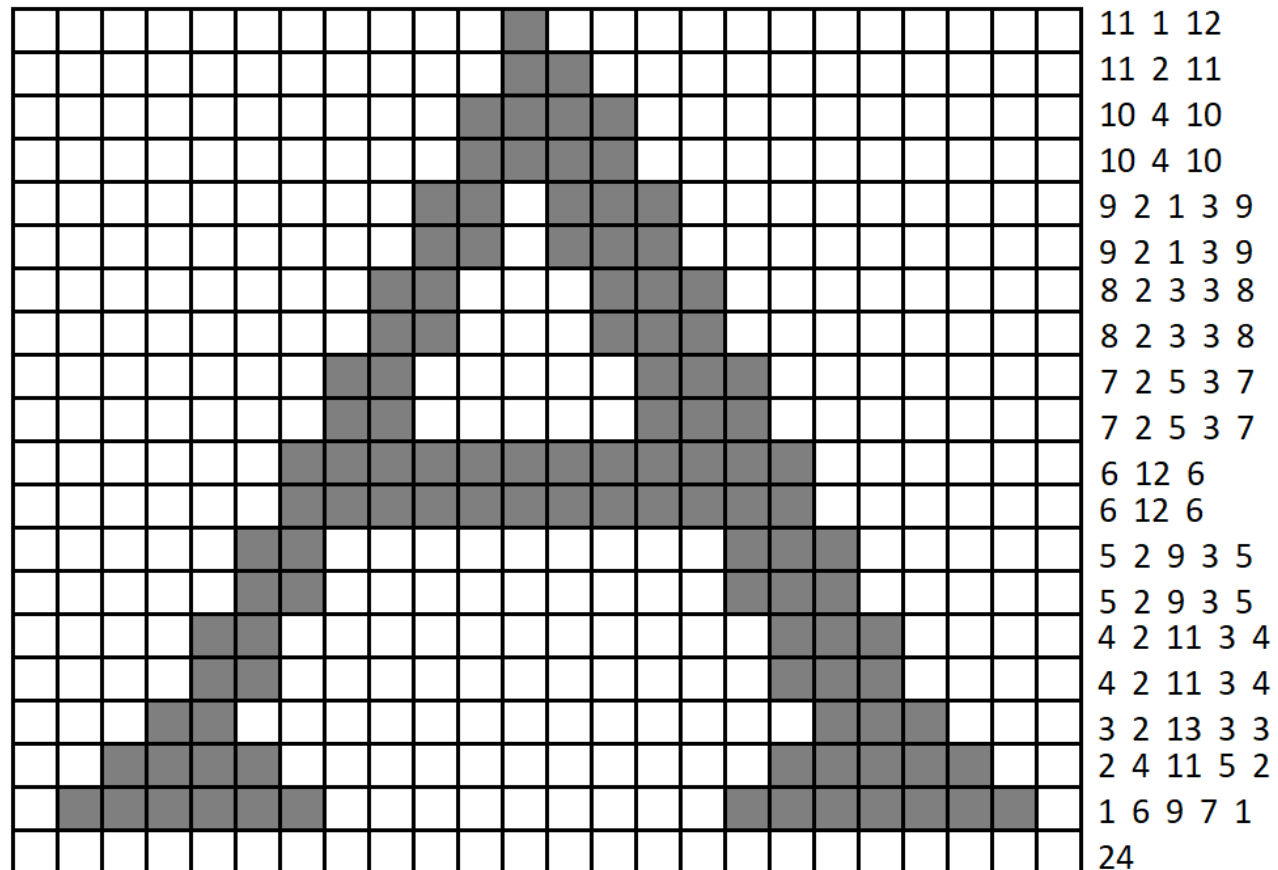
- Seria representada por:

A8B4C10B2A1B3

- Evidentemente, esta representação só se torna útil para casos em que o número de símbolos é baixo, surgindo, com elevada probabilidade, muitos símbolos repetidos ao longo dos dados.

## 2. Codificação RLE (Run-Length Encoding)

- Uma aplicação onde esta técnica tem bastante utilidade é na codificação de imagens monocromáticas (imagens a preto-e-branco), como é o caso do fax.



20x24=480 bits

85 nibbles = 340 bits

### 3. Codificação Baseada em Dicionário

---

- Uma classe de codificadores, usualmente usados na compressão de texto (mas não só), considera a existência de um dicionário de sequências símbolos (conhecido na compressão e decodificação).
- Essas sequências de símbolos (palavras) estão identificadas no dicionário através de um número (índice ou endereço).
- Na compressão (codificação), à medida que as palavras vão sendo sequencialmente encontradas no texto, são representadas pelo índice correspondente no dicionário.
- Os códigos podem considerar um dicionário fixo, e podem ir criando o dicionário à medida que codificam/decodificam o texto.

### 3. Codificação Baseada em Dicionário

---

#### **Codificador LZ77:**

- Abraham Lempel e Jacob Ziv propuseram, em 1977, um codificador que cria dinamicamente o dicionário considerado – este ficou conhecido por código (ou compressão) LZ77.
- Este método (juntamente com o LZ78) é reconhecido como um marco na técnica de compressão de dados e ficheiros, tendo sido baseada na observação de que num texto é frequente ocorrerem repetições de sequências de caracteres.
- Os algoritmos LZ77 e LZ78 foram usados como base para formar técnicas de compressão de dados que hoje estão muito disseminadas, como o GIF e o algoritmo DEFLATE (que é usada nas codificações PNG e ZIP).

# 3. Codificação Baseada em Dicionário

---

## Codificador LZ77:

- O algoritmo LZ77 é:
  - Considere-se uma janela deslizante de  $N$  caracteres em sequência (usualmente,  $N = 2\text{ kB}, 4\text{ kB}$  ou  $32\text{ kB}$ ), do próprio texto a comprimir.
  - Uma outra janela, denominada janela de observação, com  $M$  caracteres, é usada para pesquisar sequências de caracteres que existam na janela deslizante.
  - Se existir tal sequência, ela é substituída no texto comprimido pelo **par distância-comprimento** referente à janela deslizante.

### 3. Codificação Baseada em Dicionário

---

#### Codificador LZ77:

- Exemplo ilustrativo (com  $N = 8$  e  $M = 4$ ):
  - Sequência a codificar:

AEABCEHCDABCHDEABCD

---

- Primeiro, copia-se a primeira janela deslizante (dicionário inicial):

AEABCEHC

- Depois, considera-se a janela de observação que se lhe segue (DABC) e tenta-se encontrar a máxima sequência comum na janela deslizante.
- Esta será ABC, que não inclui o primeiro carater (que é copiado para a mensagem comprimida).



### 3. Codificação Baseada em Dicionário

---

#### Codificador LZ77:

- Exemplo ilustrativo (com  $N = 8$  e  $M = 4$ ):

- Sequência a codificar:

AEABCEHCDABCHDEABCD

---

- O sequência comprimida fica, neste ponto:

AEABCEHCD

- E a janela deslizante avança um carater (EABCEHCD).
- A nova janela de observação é, então: ABCH.
- A sequência ABC está na janela deslizante, começando na posição 2. Logo, a sequência comprimida fica, agora:

AEABCEHCD(2,3)

### 3. Codificação Baseada em Dicionário

---

#### Codificador LZ77:

- Exemplo ilustrativo (com  $N = 8$  e  $M = 4$ ):

- Sequência a codificar:

AEABCEHCDABCHDEABCD

---

- A janela deslizante avança 3 caracteres: CEHCDABC
- Assim, como a janela de observação: HDEA
- Como não há sequências coincidentes na janela deslizante, copia-se o próximo carater:

AEABCEHCD(2,3)H

- E as janelas avançam mais um carater:
- EHCDABCH e DEAB (e assim sucessivamente).

### 3. Codificação Baseada em Dicionário

---

#### Codificador LZ77:

- Exemplo ilustrativo (com  $N = 8$  e  $M = 4$ ):
  - Sequência a codificar:

AEABCEHCDABCHDEABCD

---

- | Sequência:                | J. deslizante: | J. de obs.: |
|---------------------------|----------------|-------------|
| • AEABCEHCD(2,3)HD        | HCDABCHD       | EABC        |
| • AEABCEHCD(2,3)HDE       | CDABCHDE       | ABCD        |
| • AEABCEHCD(2,3)HDE(3,3)  | BCHDEABC       | D           |
| • AEABCEHCD(2,3)HDE(3,3)D |                |             |

# 3. Codificação Baseada em Dicionário

---

## Codificador LZ77:

- Muitas vezes, o índice indicado não é referente à posição inicial da janela deslizante mas sim à posição do carater que se está analisar.
- Ou seja, no par distância-comprimento, a distância é interpretada como o número de caracteres que precedem o carater em análise.
- Se a dimensão da janela deslizante é pequena, torna-se fácil perder repetições de sequências que estejam um pouco afastadas no texto.
- Se essa dimensão é elevada, os índices de distância requerem um número elevado de bits (retirando eficiência à compressão).

### 3. Codificação Baseada em Dicionário

---

#### **Codificador LZW (Lempel-Ziv-Welch):**

- Trata-se de uma versão melhorada do LZ78, proposta por Lempel, Ziv e Terry Welch em 1984.
- Este considera logo desde o início a presença de um dicionário.

#### Algoritmo:

- Constitui-se o dicionário com todos os símbolos do código.
- Começa-se a percorrer os dados até que a presente sequência  $w$  mais o carácter seguinte não esteja no dicionário.
- Adiciona-se  $w$  mais o carácter seguinte ao dicionário (o dicionário vai aumentando de tamanho).
- Sequências encontradas no dicionário são substituídas pelo índice correspondente.

# 3. Codificação Baseada em Dicionário

## Codificador LZW (Lempel-Ziv-Welch):

Exemplo:

TOBEORNOTTOBEORTOBEORNOT#

Dicionário inicial:

Símbolo	Índice
#	0
A	1
B	2
...	...
Z	26

5 bits por símbolo



- Início da análise dos dados: a sequência TO não está no dicionário, pelo que se adiciona ao dicionário.

TO	27
O...	28

### 3. Codificação Baseada em Dicionário

---

#### Codificador LZW (Lempel-Ziv-Welch):

Exemplo:

TOBEORNOTTOBEORTOBEORNOTTO#

- Na sequência de saída coloca-se o índice correspondente ao símbolo T.
- Testa-se, agora a sequência seguinte: OB
- Também não está no dicionário, pelo que é adicionada (e o índice de O é enviado para a sequência de saída).

Dicionário:

TO	27
OB	28
B...	29

### 3. Codificação Baseada em Dicionário

#### Codificador LZW (Lempel-Ziv-Welch):

Exemplo:

TOBEORNOTTOBEORTOBEORNOT#

- Próxima sequência: BE (adiciona-se ao dicionário, e B para a saída).

Dicionário:

TO	27
OB	28
BE	29
E...	30

- Próxima: EO (para o dicionário)  
(E para a saída)

TO	27
OB	28
BE	29
EO	30
O...	31



### 3. Codificação Baseada em Dicionário

#### Codificador LZW (Lempel-Ziv-Welch):

Exemplo:

TOBEORNOTTOBEORTOBEORNOT#

- Próxima sequência: OR (para ao dicionário, e O para a saída).

Dicionário:

TO	27
OB	28
BE	29
EO	30
OR	31
R...	32

A partir daqui são necessários  
6 bits por símbolo

- Próxima: RN (para o dicionário) ETC.

### 3. Codificação Baseada em Dicionário

---

#### Codificador LZW (Lempel-Ziv-Welch):

Exemplo:

TOBEORNOTTOBEORTOBEORNOT#

- Quando chega ao 10º caractere, encontra-se a primeira sequência no dicionário: coloca-se o índice 27 na saída.
- Adiciona-se TOB ao dicionário.

E continua-se o procedimento até ao final dos dados.

Naturalmente, o dicionário não pode crescer indefinidamente.

TO	27
OB	28
BE	29
EO	30
OR	31
...	...
TT	35
TOB	36
B...	

### 3. Codificação Baseada em Dicionário

#### Codificador LZW (Lempel-Ziv-Welch):

Exemplo:

TOBEORNOTTOBEORTOBEORNOT#

- No final, a sequência de saída fica:

20	15	2	5	15	18	14	15	20	27	29	31	36	30	32	34	0
T	O	B	E	O	R	N	O	T	TO	BE	OR	TOB	EO	RN	OT	#
5	5	5	5	5	5	6	6	6	6	6	6	6	6	6	6	6

- Número total de bits:  $6 \times 5\text{bit} + 11 \times 6\text{bit} = 96 \text{ bits}$
- Antes da compressão:  $25 \times 5\text{bit} = 125 \text{ bits}$
- Reduz-se em cerca de 22% o volume de dados.

## 4. Codificadores para Imagem sem Perdas

---

- Existem vários algoritmos de compressão, sem perda de informação, destinados à compressão de imagens.
- Alguns exemplos são:
  - **JBIG**: compressão para imagens binárias (fax, por exemplo) – implementa RLE, seguido de codificação de Huffman.
  - **JPEG-LS**: Usa predição linear entre pixéis adjacentes e codificação de Huffman sobre o erro.
  - **GIF**: Usa o código LZW, mas restringe a paleta de cores ao máximo de 256.
  - **PNG**: Usa predição linear e o algoritmo DEFLATE (que se baseia no LZ77 e no LZW) para comprimir o erro. Admite “*true color*”.