Ana Rita Almeida - 35456, Pedro Sousa - 38130

UFP | 2021-2022

*Algoritmo Run Length Encoding (RLE)*

Multimédia II

Índice

[Resumo 2](#_Toc104902707)

[1. Introdução 3](#_Toc104902708)

[1.1. Objetivos do trabalho 3](#_Toc104902709)

[1.2. Justificação 3](#_Toc104902710)

[1.3. Áreas de aplicação 3](#_Toc104902711)

[2. O algoritmo Run Length Encoding 4](#_Toc104902712)

[3. Implementação do algoritmo Run Length Encoding 5](#_Toc104902713)

[3.1. Compressão de texto 5](#_Toc104902714)

[3.2. Descompressão de texto 6](#_Toc104902715)

[3.3. Compressão de imagem 7](#_Toc104902716)

[3.4. Descompressão de imagem 8](#_Toc104902717)

[4. Comparação com outros métodos 9](#_Toc104902718)

[5. Corpus Silesia 10](#_Toc104902719)

[6. Conclusão 11](#_Toc104902720)

[Bibliografia 12](#_Toc104902721)

# Resumo

Este artigo científico foi desenvolvido para a disciplina de Multimédia II e consiste em estudar o método *Run Length Encoding* e explorar os seus algoritmos para codificação e descodificação de texto e imagens.

# Introdução

O *Run Length Encoding* é um processo de compressão de caracteres quando existe uma longa sequência dos mesmos (4 ou mais). É um algoritmo considerado simples e o seu *codec* (codificador/descodificador) são sem perdas e supressão de sequências repetitivas.

## Objetivos do trabalho

## 

O objetivo deste trabalho é desenvolvermos uma aplicação de software que implemente o algoritmo RLE. O trabalho será maioritariamente realizado em *JavaScript*, com HTML e CSS para ficar disponível em versão Web.

## Justificação

Entre os elementos do grupo, achamos que o RLE seria uma boa opção. Embora este método não seja dos mais eficientes (devido às restrições dos números de repetições de caracteres), é um algoritmo muito utilizado e também cumpre o requisito 1 que impõe a implementação de um algoritmo de compressão/descompressão **sem perdas**.

## Áreas de aplicação

Este *codec* é usado para a compressão de texto e imagens, especialmente em áreas onde não se pode perder informação pois estes métodos são sem perdas. No final dos anos 60 era utilizado na transmissão de sinal televisivo e, na década de 80, foi introduzido pela *CompuServe*, um serviço online que disponibiliza conexão à internet, para comprimir imagens partilhadas na rede. Atualmente o RLE é utilizado em imagens *“BMP”*, no sistema operativo *Windows*, e em imagens *“JPG”*, juntamente com a transformada discreta de cosseno.

# O algoritmo *Run Length Encoding*

*Run Length Encoding* é um método de compressão sem perdas de dados onde uma determinada sequência de valores repetidos são combinados como um único valor e a quantidade de valores repetidos no lugar da sequência original.

Este algoritmo é especialmente útil e eficaz para grandes conjuntos de repetições.

# Implementação do algoritmo *Run Length Encoding*

## Compressão de texto

O processo de codificação por RLE é o seguinte:

* Determinar uma *Flag* que não exista no texto que vai ser comprimido, como um caractere especial por exemplo.
* Em seguida, vamos ler o caractere na posição i (primeira posição, a inicial). Se o caractere seguinte (i+1) não for igual ao da posição i (texto[i+1] != texto[i]), i+1 passa a ser o novo i e começamos novamente a sequência. Se i+1 for igual a i, vamos verificando os caracteres seguintes (i+2, i+3, etc.) enquanto os caracteres nessas posições sejam iguais ao caractere na posição i (texto[i]).
* Quando essa sequência terminar vamos contar o número de vezes que o caractere se repetiu.
* Se a repetição for menor que 4, não se efetua compressão e então os caracteres vão permanecer iguais.
* Se a sequência de caracteres repetidos for igual ou maior que 4, faz-se a compressão e o resultado será algo como: *Flag* + nº de repetições + caracter.

Exemplo:

Vamos selecionar a *Flag* “$”

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A | A | A | B | B | B | C | C | C | C | C | D | D | D | D | D | D |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| $ | 4 | A | B | B | B | $ | 5 | C | $ | 6 | D |

## Descompressão de texto

O processo de descodificação por RLE é o seguinte:

* Determinar a *Flag* que está a ser usada;
* Criar uma *string* auxiliar vazia.
* Ler o texto até a *Flag* ser encontrada.
* Ao encontrar a *Flag*, iremos estar na posição texto[i]. Na posição

texto[i + 1] vamos encontrar o número de repetições de um caractere e na posição texto[i + 2] vamos encontrar o caractere repetido.

* Em seguida, criamos uma string de repetições com o tamanho do número de repetições do caractere e colocamos esse caractere em todas as posições.
* Vamos fazer a concatenação da *string* auxiliar que criamos no início com a string de repetições. E depois zeramos a *string* de repetições para voltar a usar porque cada vez que aparecer a *Flag* na *string* texto, vamos repetir o ponto anterior, seguido deste ponto.

Exemplo:

*Flag*: #

i i+1 i+2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # | 5 | A | B | B | # | 6 | C |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | A | A | A | A |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B | B | # | 6 | C |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | A | A | A | A | B | B |

|  |  |  |
| --- | --- | --- |
| # | 6 | C |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | A | A | A | A | B | B | C | C | C | C | C | C |

## Compressão de imagem

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

* Temos aqui uma imagem de 7 por 6 pixéis.
* Vamos analisar linha a linha e agrupar os pixéis contíguos que possuam a mesma cor:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 3 | 1 | 1 |  |  |
| 1 | 2 | 1 | 2 | 1 |  |  |
| 1 | 1 | 1 | 1 | 1 | 2 |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 3 | 2 |  |  |  |  |
| 3 | 1 | 3 |  |  |  |  |

Resultado:

1W1R3W1B1W

1G2O1W2O1P

1Y1P1B1Y1G2Y

1W1G1R1G1P1G1W

2W3B2W

3W1P3W

## Descompressão de imagem

A descompressão de imagens é efetivamente o processo simétrico da compressão de imagem.

# Comparação com outros métodos

Comparamos o RLE com a codificação de *Huffman* e a codificação aritmética e concluímos que a vantagem que o RLE tem em relação a estes dois é que é fácil de implementar e consome muita pouca capacidade de CPU, no entanto, em imagens, o RLE consegue ser eficiente porque existem muitos pixéis iguais e consegue uma taxa de compressão muito favorável. No entanto, não é dos mais eficientes para linguagem natural porque é difícil encontrar palavras que tenham uma letra várias vezes repetida consecutivamente.

# Corpus Silesia

O Algoritmo RLE não é recomendado nem eficiente para exemplos como o Corpus Silesia porque no vocabulário é muito invulgar (ou impossível mesmo) haver palavras com 4 ou mais caracteres repetidos sucessivamente para haver compressão!

Uma imagem com texto

Descrição gerada automaticamente

Figura - Compressão do ficheiro "Dickens" do Corpus Silesia

# Conclusão

Após a realização deste trabalho, podemos concluir que o algoritmo de compressão *Run-Length Encoding* é relativamente simples e eficiente quando se trata de imagens. No que toca a texto, tal como foi referido, não é muito eficiente quando usado para linguagem natural. Com a implementação deste algoritmo, ganhámos mais conhecimentos sobre a linguagem de programação *JavaScript*.

# Bibliografia

* Salomon, D., “Data Compression The Complete Reference”, Springer, 4th. Edition, 2007
* <http://multimedia.ufp.pt/codecs/compressao-sem-perdas/supressao-de-sequencias-repetitivas/run-length-encoding/>