Algoritmos de compressão

Prof. Dieisson Martinelli

dieisson.martinelli@udesc.br

Programa

- Introdução
- Frequência de caracteres
- Algoritmo de Huffman
- Algoritmo LZW
- Atividades

Introdução

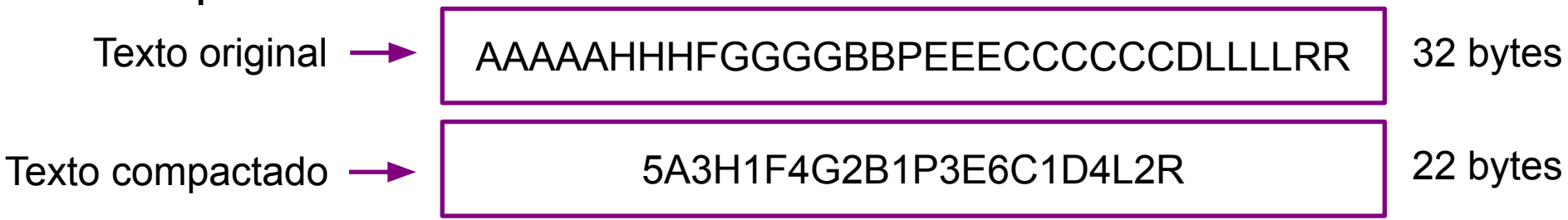
- A **compressão de dados** (*data compression*) consiste na utilização de um conjunto de métodos p/ **reduzir o tamanho** dos **dados** (arquivos) com o intuito de "economizar" **espaço de armazenamento** em unidades de memória de um sistema computacional
  - Um **arquivo comprimido** terá seu **tamanho reduzido** com a aplicação de um algoritmo de compressão (ou compactação). Tal como a função dos programas ***gzip***, ***winzip*** ou ***winrar***
  - Esses programas também incluem algoritmos de empacotamento, permitindo que **múltiplos arquivos** sejam **compactados** e **concatenados** dentro de um único arquivo

Introdução

- Muitos arquivos de extensões bem conhecidas, como **PDF** (textos, *e-books*), **MP3** (áudio, músicas), **GIF** (imagens, fotos), **ZIP** (arquivos em geral), **MPG** (vídeos), utilizam algoritmos de compactação em suas concepções
  - Diversos desenvolvedores e pesquisadores criaram (e continuam criando) **algoritmos de compressão** de arquivos, com finalidade de uso nas mais diversas áreas da computação
- Alguns algoritmos de compactação:
  - Frequência de caracteres (conceitual)
  - Algoritmo de Huffman (David A. Huffman, 1952)
  - Algoritmo LZW (A. Lempel, J. Ziv, T. Welch, 1984)

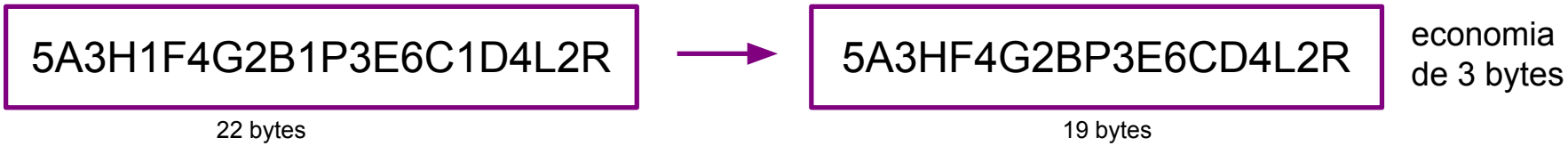
Frequência de Caracteres

- É um algoritmo que possibilita **compactar arquivos** contendo **texto alfabético**
  - Procura determinar a quantidade de **símbolos idênticos consecutivos** existentes no texto
  - Cada sequência de símbolos idênticos do texto é substituída por um **número** indicando a **frequência do símbolo** em questão
- Exemplo:

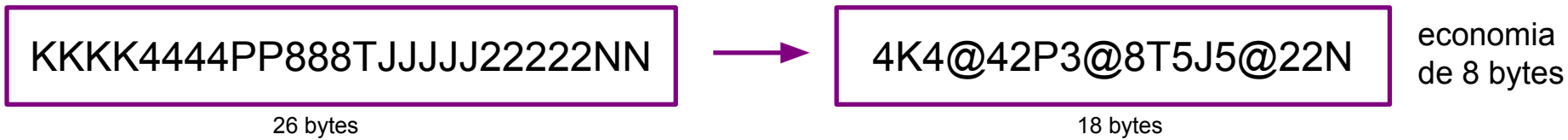


Frequência de Caracteres

- A compactação ainda poderia ser melhorada se a **ausência do número** que indica a **frequência** implique em **frequência 1**



Outra melhoria, considerando um **texto alfanumérico**, seria a inclusão do **caractere "@"** para indicar que será apresentado um **símbolo do texto original** e não uma frequência



Algoritmo de Huffman

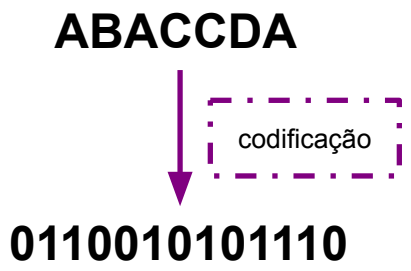
- O **algoritmo** (ou codificação) **de Huffman** segue o mesmo princípio do algoritmo de frequência de caracteres, mas utilizando uma árvore binária
  - Ao invés de expressar a frequência de caracteres na compactação, o texto original é convertido em um **texto codificado somente por bits** (0s e 1s)
- Exemplo p/ um alfabeto A, B, C e D:
  - O texto **ABACCDA** seria codificado em **01010001000000111010**
    - Passando de 56 bits para 21

Símbolo	Código
A	010
B	100
C	000
D	111

Algoritmo de Huffman

- No exemplo anterior, pode-se obter uma **codificação menor** se para cada símbolo forem associados os seguintes códigos:

Símbolo	Código
A	0
B	110
C	10
D	111



- A mensagem codificada a partir do texto original agora ocupa de somente 13 bits

Algoritmo de Huffman

- No exemplo anterior, a **codificação** foi capaz de **diminuir o tamanho** do texto porque o **menor código**, representando a letra A, **aparece** com **mais frequência** que os códigos das letras B e D
- A ideia por trás do algoritmo de Huffman está em **representar os símbolos** que ocorrem com maior frequência com **menos bits** e os que ocorrem com menor frequência com **mais bits** (8, por exemplo)
- **Lembrando**: para representar cada caractere binariamente, são necessários 8 bits ($8^2 = 256$) – Tabela ASCII

Algoritmo LZW

- **LZW** (Lempel-Ziv-Welch) é um **algoritmo de compressão** cujo nome é derivado dos nomes de seus desenvolvedores: Abraham **L**empel, Jakob **Z**iv e Terry **W**elch
- **Característica**: busca substituir sequências de símbolos (por exemplo, caracteres) por códigos
  - Para obter a compressão de dados, os códigos devem ser menores que as sequências representadas por eles
- O algoritmo LZW é usado para compactar arquivos de texto e arquivos binários em geral, como imagens e vídeos

Algoritmo LZW - Codificação

- Inicializar o dicionário (com símbolos básicos).
- Repetir até o fim do texto:
  - A partir da posição atual, achar a maior string **w** existente no dicionário
  - Escrever o índice de **w** na saída
  - Olhar o próximo caráter a que não fez parte de **w**
  - Escrever **wa** no dicionário
  - Avançar para a posição de **a**

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	
4	
5	
6	
7	
8	
9	

w a b b a w a b b a

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	
5	
6	
7	
8	
9	

w a b b a w a b b a

2

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	
6	
7	
8	
9	

w a b b a w a b b a

2 0

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	
7	
8	
9	

w	a	b	b	a	w	a	b	b	a
2	0	1							

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	
8	
9	

w	a	b	b	a	w	a	b	b	a
2	0	1	1						

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	aw
8	
9	

w	a	b	b	a	w	a	b	b	a
2	0	1	1	0					

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	aw
8	wab
9	

w	a	b	b	a	w	a	b	b	a
2	0	1	1	0	3				

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	aw
8	wab
9	bba

w	a	b	b	a	w	a	b	b	a
2	0	1	1	0	3		5		0

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	aw
8	wab
9	bba

w	a	b	b	a	w	a	b	b	a
2	0	1	1	0	3		5		0
0010	0000	0001	0001	0000	0011		0101		0000

8 números de 4 bits = 32 bits

00100000001000100000011010100000

Compressão = 1 - 32/80 = 60%

Algoritmo LZW - Decodificação

- Inicializar o dicionário (com símbolos básicos).
- Decodificar o 1º índice, escrevê-lo na saída e armazená-lo em **w**
- Colocar **w?** no dicionário
- Repetir até o fim dos índices:
  - Decodificar o primeiro simbolo s do próximo índice
  - Trocar o **?** da última entrada no dicionário por **s**
  - Decodificar o resto do índice, escrevê-lo na saída e armazená-lo em **w**
  - Colocar **w?** no dicionário

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	
4	
5	
6	
7	
8	
9	

2 0 1 1 0 3 5 0

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	w?
4	
5	
6	
7	
8	
9	

2 0 1 1 0 3 5 0

w

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	a?
5	
6	
7	
8	
9	

2 0 1 1 0 3 5 0

w a

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	b?
6	
7	
8	
9	

2 0 1 1 0 3 5 0

w a b

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	b?
7	
8	
9	

2	0	1	1	0	3	5	0
w	a	b	b				

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	a?
8	
9	

2	0	1	1	0	3	5	0
w	a	b	b	a			

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	aw
8	wa?
9	

2	0	1	1	0	3	5	0
w	a	b	b	a	wa		

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	aw
8	wab
9	bb?

2	0	1	1	0	3	5	0
w	a	b	b	a	wa	bb	

Algoritmo LZW - Codificação

0	a
1	b
2	w
3	wa
4	ab
5	bb
6	ba
7	aw
8	wab
9	bba

2	0	1	1	0	3	5	0
w	a	b	b	a	wa	bb	a

Atividades

1. Implementar a compactação e descompactação de arquivos para o algoritmo LZW.

*Enviar as atividades no Moodle

Trabalho de Pesquisa

- Explorar algoritmos , técnicas ou padrões **(diferentes dos vistos em aula)** utilizados para compactar dados, seja em imagens, vídeos, textos, áudio ou outros tipos de informação, destacando **como funcionam**, **quais problemas resolvem** e **onde são aplicados**.

- Apenas apresentação! (15 minutos)
- Dia 16/04/2025: Aula livre para pesquisa!
- Apresentação no dia: 23/04/2025