

Técnicas de compressão não destrutiva

Teoria da informação



Gabriel Fernandes

Maria Dias

Pedro Rodrigues

TÓPICOS

① Introdução

② Etapas de codificação

③ Algoritmos usados

④ Combinações mais usuais

⑤ Conclusão

TÓPICOS

① Introdução

② Etapas de codificação

③ Algoritmos usados

④ Combinações mais usuais

⑤ Conclusão

Introdução

Necessidade de resolver
problemas relacionados com a
transferência de informação



**Canais existentes não atingem
velocidades aceitáveis**

**Larguras de banda pequenas para a
quantidade de informação a
transferir**

**Necessário diminuir o tamanho da
informação**

Compressão não destrutiva

- Comprime sem qualquer perda de informação
- Baseado em técnicas que visam a reorganização da fonte de dados

Compressão destrutiva

- Leva à perda de informação
- Remove bits desnecessários para reduzir o tamanho do ficheiro

COMPRESSÃO É POSSÍVEL DEVIDO:

- Redundância;
- Dependência estatística dos símbolos do ficheiro fonte

TÓPICOS

① Introdução

② Etapas de codificação

③ Algoritmos usados

④ Combinações mais usuais

⑤ Conclusão

Etapas de codificação

Compressão

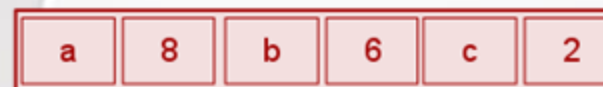


É possível
devido

- **Redundância**
- **Dependência estatística**



run-length encoding



Escolha de algoritmo de compressão

- Tempo de compressão

$$\text{Compression speed} = \frac{\text{Uncompressed file size}}{\text{Time to Compress}}$$

- Tempo de descompressão

$$\text{Decompression speed} = \frac{\text{Compressed file}}{\text{Time to Decompress}}$$

- Taxa de compressão

$$\text{Compression ratio} = \frac{\text{Uncompressed file size}}{\text{Compressed file size}}$$

Etapas de codificação



Transformação da fonte

(através de algoritmos como Predictor e BW);

Análise da dependência estatística e tratamento de dados

(codificação por dicionário, modelação através de cadeias de Markov e Run Length Encoding)

Representação de agrupamentos utilizando técnicas de codificação entrópica

(codificação aritmética e árvores de Huffman)

TÓPICOS

① Introdução

② Etapas de codificação

③ **Algoritmos usados**

④ Combinações mais usuais

⑤ Conclusão

Algoritmos usados

PREDICTOR

Transformação da fonte através de previsões e de relações estatísticas entre elementos adjacentes da mesma fonte

01

BW (Burrows – Wheeler)

Aumentar a redundância da fonte de informação

02

1ª Etapa

Redundância da fonte



PREDICTOR

Transformação da fonte através de previsões e de relações estatísticas entre elementos adjacentes da mesma fonte

01

Delta Encoding

É uma das técnicas utilizadas por algoritmos de LPC (Linear Predictive Coding).

Podemos verificar que esta técnica utilizada neste tipo de algoritmos nos permite de certa forma quantizar a fonte e tirar partido das relações entre os elementos! Obtemos assim uma grande eficiência aplicando estas técnicas na codificação de imagens explorando a redundância entre pixéis.

original data stream:	17	19	24	24	24	21	15	10	89	95	96	96	96	95	94	94	95	93	90	87	86	86	...
	<i>move</i>	<i>delta</i>	<i>delta</i>	<i>delta</i>	...																		
delta encoded:	17	2	5	0	0	-3	-6	-5	79	6	1	0	0	-1	-1	0	1	-2	-3	-3	-1	0	...

Fonte: <https://www.dspguide.com/ch27/4.htm>

BW (Burrows – Wheeler)

Aumentar a redundância da fonte de informação

02

Transformation				
Input	All Rotations	Sorting All Rows into Lex Order	Taking Last Column	Output Last Column
<code>^BANANA </code>	<code>^BANANA </code> <code> ^BANANA</code> <code>A ^BANAN</code> <code>NA ^BANA</code> <code>ANA ^BAN</code> <code>NANA ^BA</code> <code>ANANA ^B</code> <code>BANANA ^</code>	<code>ANANA ^B</code> <code>ANA ^BAN</code> <code>A ^BANAN</code> <code>BANANA ^</code> <code>NANA ^BA</code> <code>NA ^BANA</code> <code>^BANANA </code> <code> ^BANANA</code>	<code>ANANA ^B</code> <code>ANA ^BAN</code> <code>A ^BANAN</code> <code>BANANA ^</code> <code>NANA ^BA</code> <code>NA ^BANA</code> <code>^BANANA </code> <code> ^BANANA</code>	<code>BNN^AA A</code>

Fonte: https://banana-slug.soe.ucsc.edu/lecture_notes:06-01-2015

- S é a fonte de informação, que contém um caracter cuja função é marcar o final de S (EOF).
- S é submetida a um total de n rotações (reversões) obtemos, assim, uma matriz de strings. Depois a matriz n por n com as strings obtidas a partir de S são ordenadas por ordem lexical ou numérica.
- Entre todas as linhas dessa matriz existirá uma igual a S. Selecionamos então a última coluna da matriz ordenada e o índice da string da matriz igual a S, o que nos permite reverter o processo posteriormente.

LZ77

Utiliza uma janela deslizante de tamanho n onde
são mantidas os últimos n símbolos
enviados/recebidos

01

LZW

Recorre a um dicionário iniciado com os
caracteres do alfabeto da fonte

02

RLE

Reduz uma sequência de símbolos iguais

03

2ª Etapa

Dependência Estatística



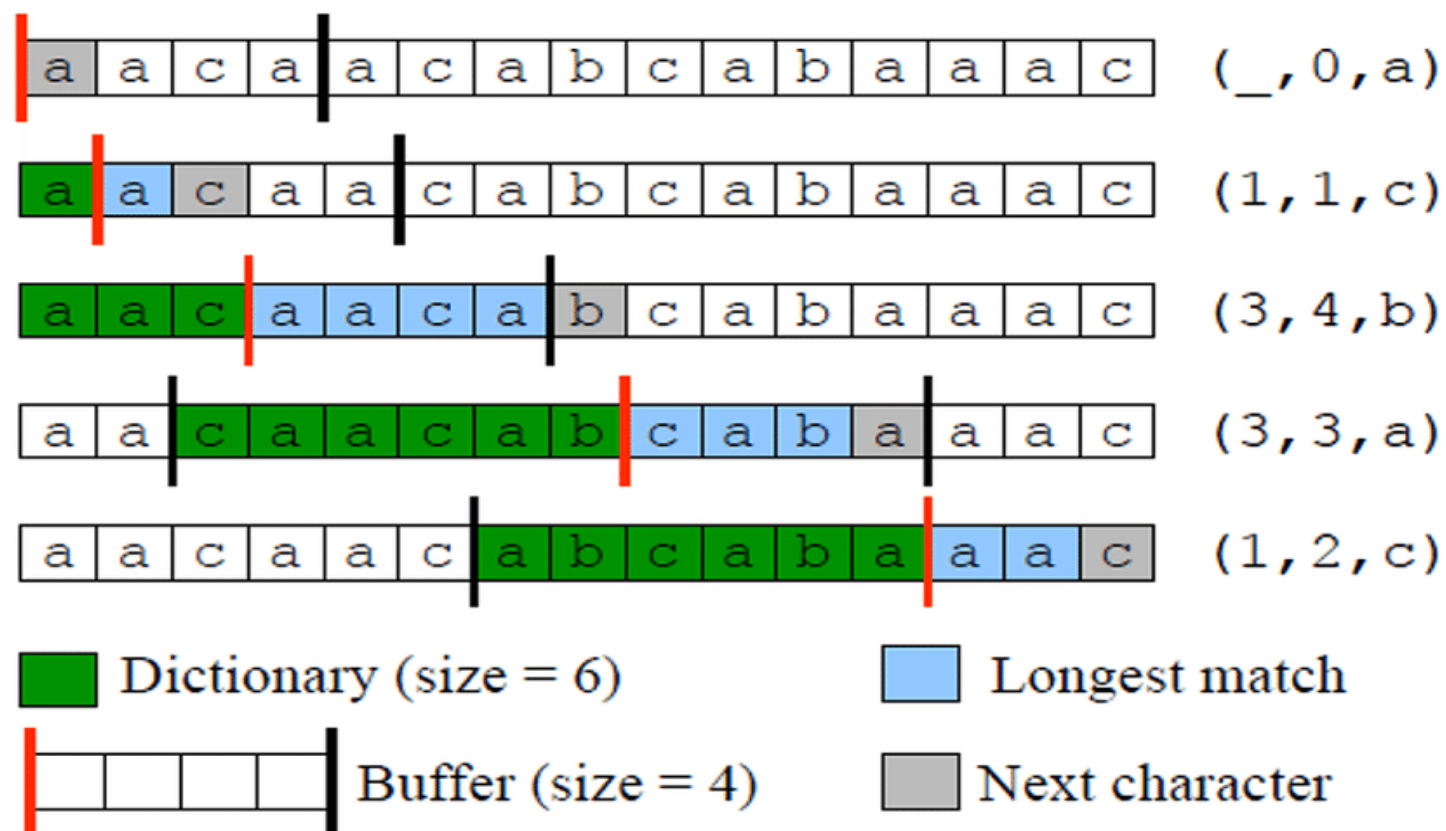
LZ77

Utiliza uma janela deslizante de tamanho n onde
são mantidas os últimos n símbolos
enviados/recebidos

01

Código na forma: (<offset>, <length>, <next symbol>)

Nota: Se não for encontrada uma sequência para copiar no search window, o
offset e a length são 0, diminuindo a eficiência do algoritmo.

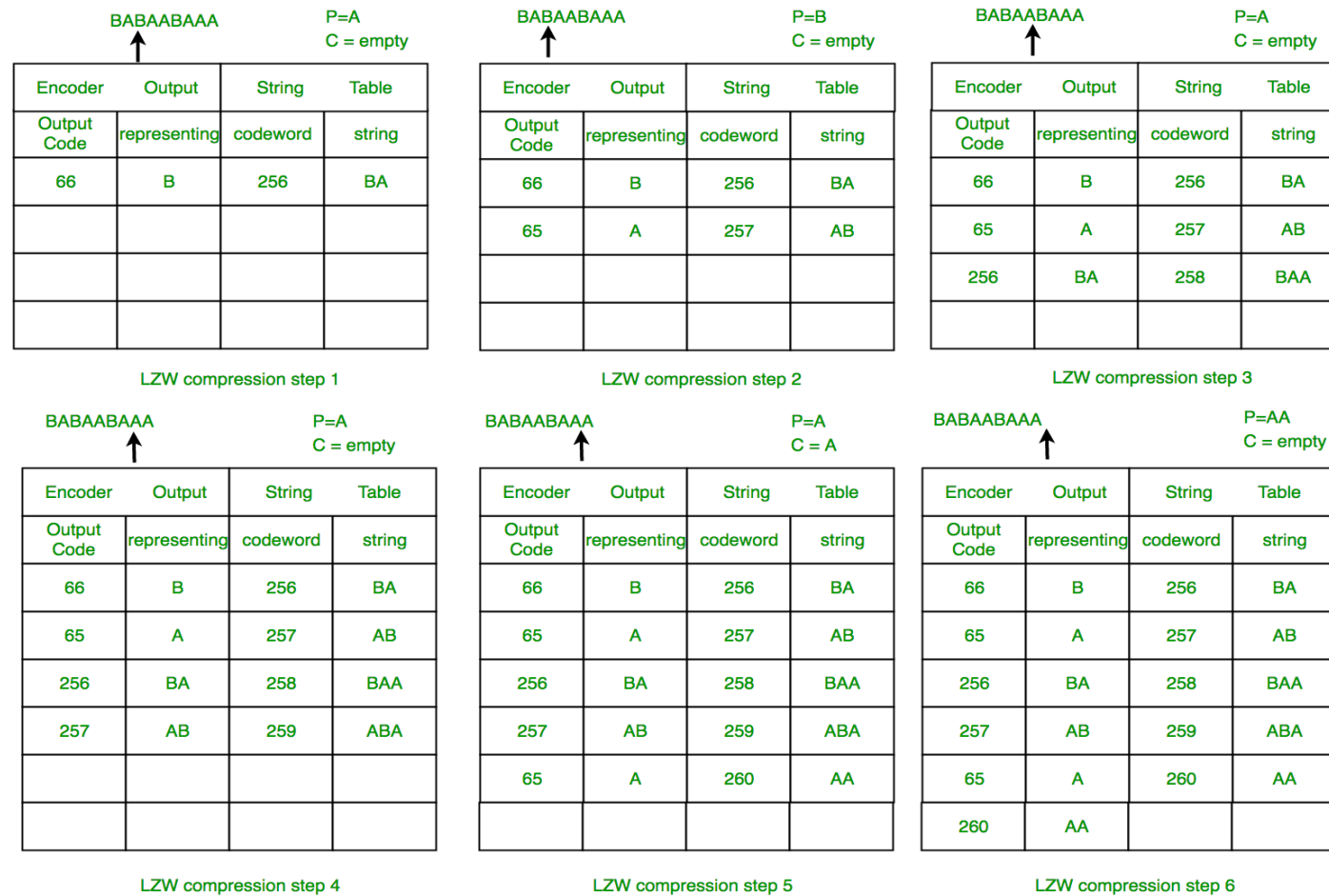


LZW

Recorre a um dicionário iniciado com os caracteres do alfabeto da fonte

02

- LZW é uma versão do LZ78, que tal como este recorre a um dicionário explícito ao contrário do seu antecessor LZ77
- O Dicionário é construído de forma adaptativa !



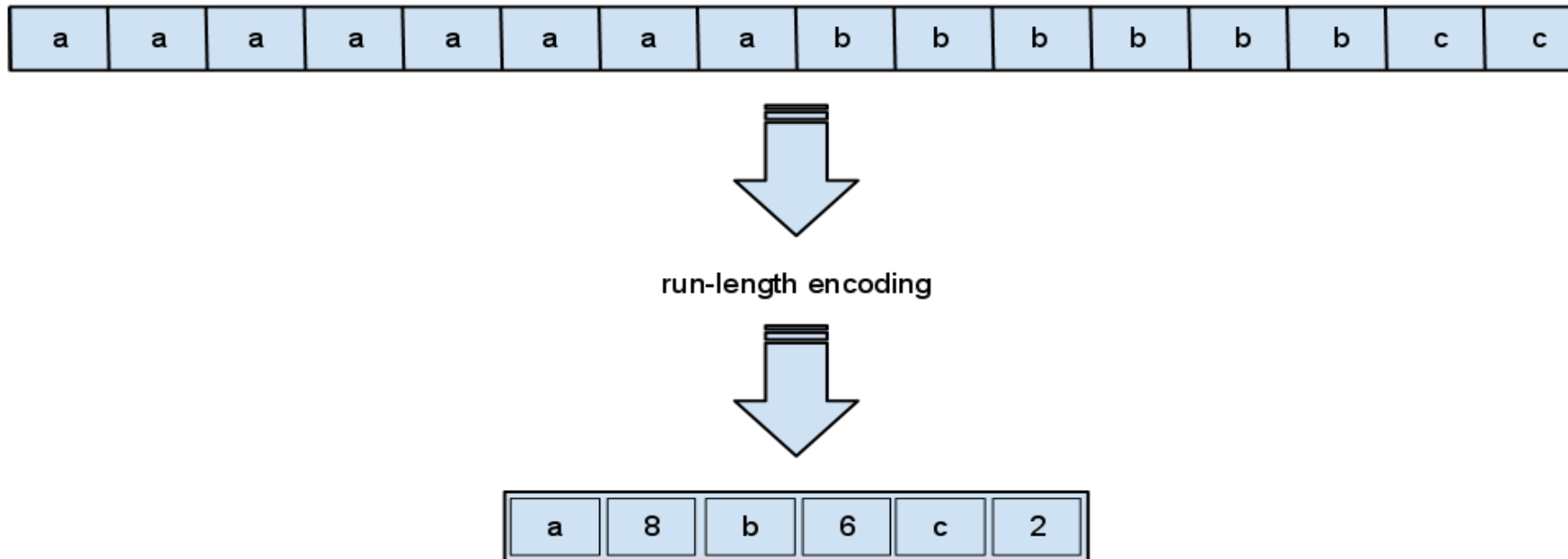
- Adiciona-se mais uma entrada no dicionário com o código da mensagem enviada + caracter seguint e da fonte de informação.
- Envia-se o índice do dicionário respetivo à melhor correspondência
- Procura-se no dicionário qual é a maior cadeia que mais caracteres têm iguais ao que se pretende enviar

RLE

03

Reduz uma sequência de símbolos iguais

- Run length encoding, consiste em reduzir uma sequência de símbolos iguais a uma sequência de 3 símbolos.
- Desses 3 símbolos fazem parte o símbolo que se repete na fonte de informação, o número de vezes que o símbolo é repetido e um símbolo especial que permite diferenciar esta forma de comprimir informação de uma sequência idêntica na fonte.



Fonte: <http://stoimen.com/2012/01/09/computer-algorithms-data-compression-with-run-length-encoding/>

Algoritmos usados

HUFFMAN CODING

Tira partido da probabilidade de ocorrência dos símbolos aos que mais se repetem são atribuídos comprimentos menores

01

ARITHMETIC CODING

Codifica a informação com base na probabilidade de cada símbolo na fonte de informação

02

3ª Etapa

Codificação entrópica

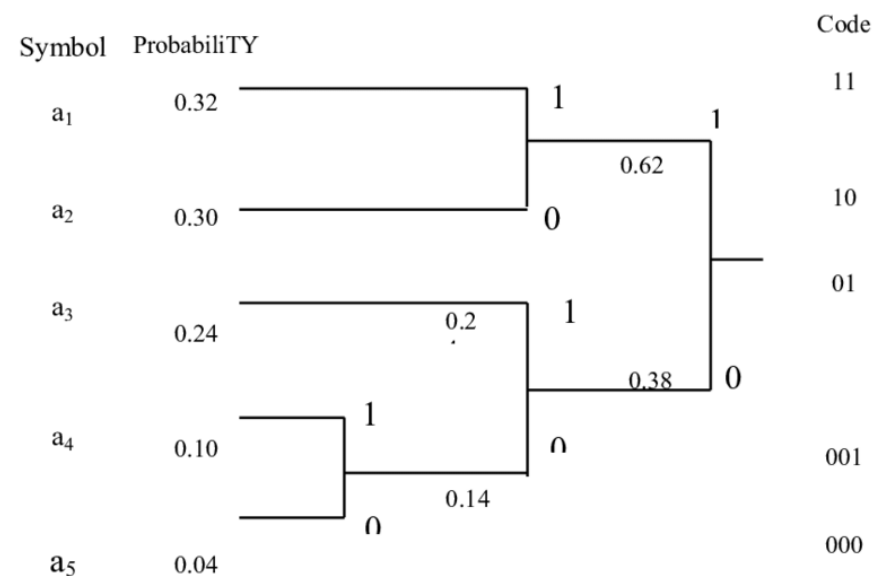


HUFFMAN CODING

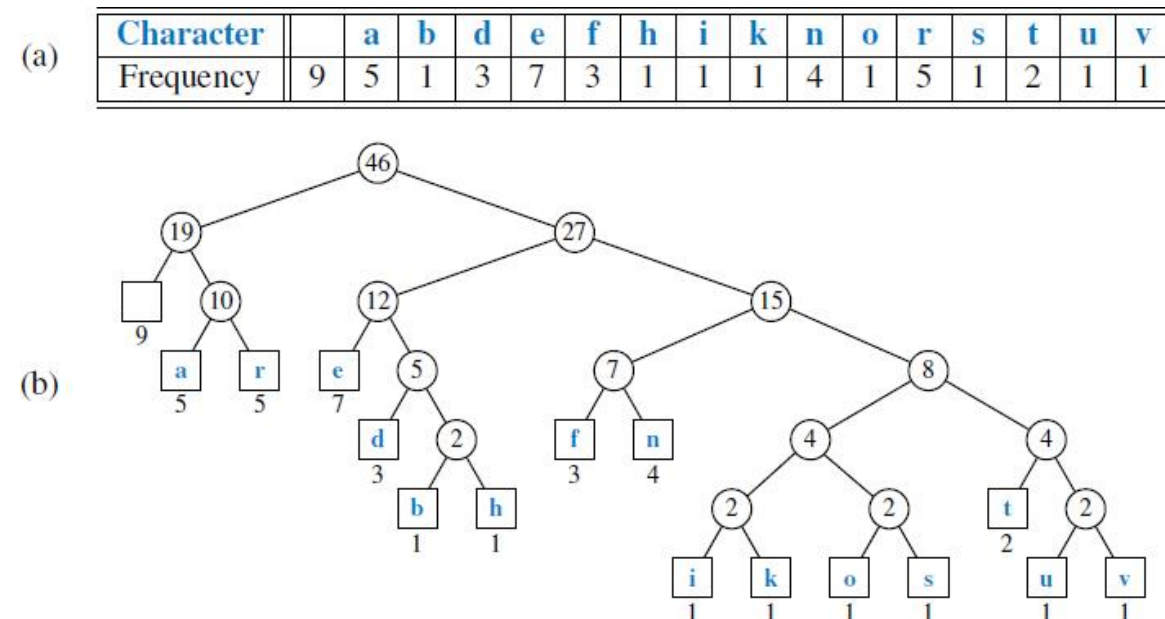
Tira partido da probabilidade de ocorrência dos símbolos aos que mais se repetem são atribuídos comprimentos menores

01

- Algoritmo que tira partido da probabilidade de ocorrência dos símbolos da fonte de informação, atribuindo aos que mais se repetem códigos de comprimento menor, enquanto que os que menos se repetem são codificados com códigos mais extensos.
- Os códigos são obtidos por via da construção de uma árvore a partir das folhas agrupando sempre os que têm menor probabilidade e, quando há mais do que dois com a mesma probabilidade e esta é a mais baixa, escolhem-se os dois que estão mais longe da raiz (assim garantimos que a variância dos códigos é mínima).



Fonte: https://www.researchgate.net/figure/An-example-of-Huffman-coding-45-Linear-Prediction-The-applications-of-the-linear_fig4_259007386



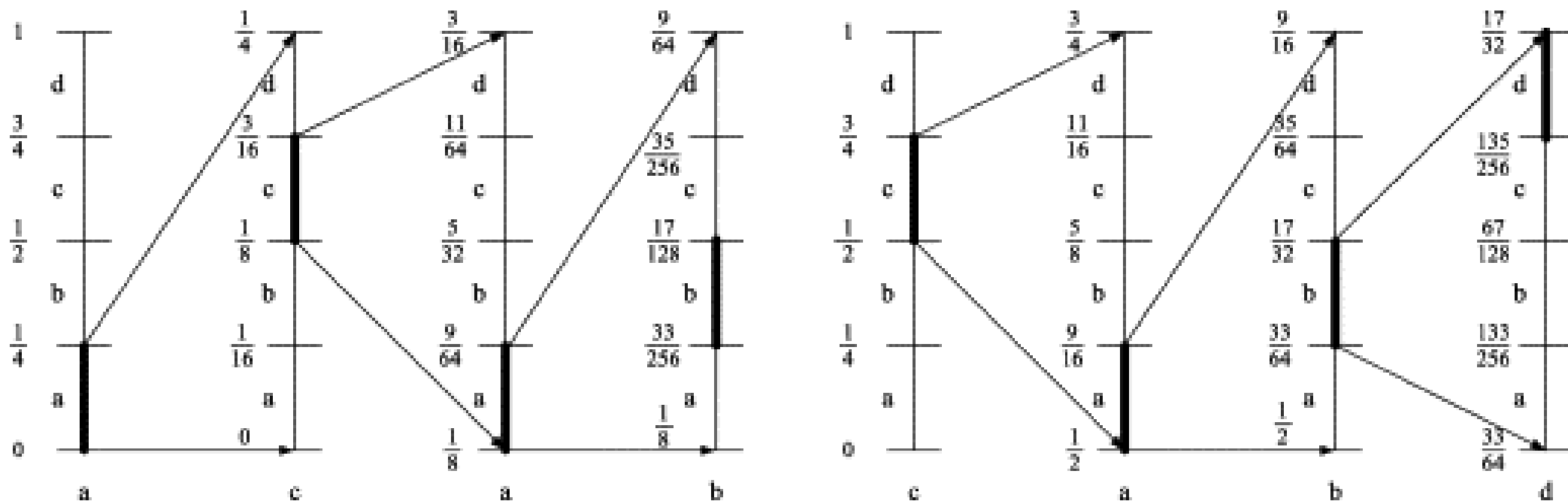
Fonte: <https://massivealgorithms.blogspot.com/2014/06/greedy-algorithms-set-3-huffman-coding.html>

ARITHMETIC CODING

Codifica a informação com base na probabilidade de cada símbolo na fonte de informação

02

- Para codificar o primeiro símbolo da fonte de informação divide-se o intervalo $[0, 1]$ pelos símbolos do alfabeto da fonte (com base na probabilidade de cada um),
- Para o segundo símbolo dividem-se todos os subintervalos criados para o símbolo anterior com base nas probabilidades de cada símbolo e assim sucessivamente.
- Basta escolher um valor que pertença a esse intervalo para codificar o símbolo da fonte, normalmente escolhe-se o valor médio.



TÓPICOS

① Introdução

② Etapas de codificação

③ Algoritmos usados

④ Combinações mais usuais

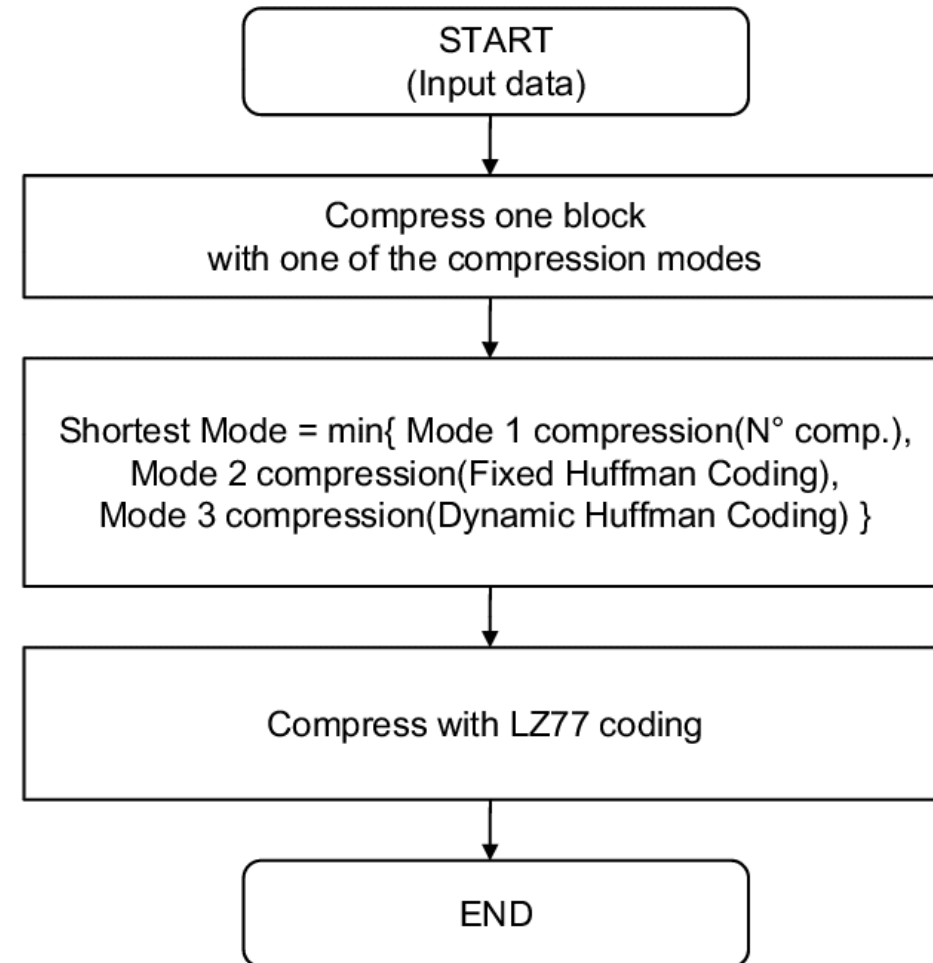
⑤ Conclusão

Deflate (Huffman Coding + LZ77)

- Associação e substituição de séries de bytes
- Substituição de símbolos tendo em conta a sua frequência de uso
- Este algoritmo é utilizado em diversos softwares de compressão de dados que tem por base o padrão ZIP ou no padrão gzip. Como por exemplo o PKZIP (Phil Katz ZIP)



Fonte: https://pkware.cachefly.net/webdocs/manuals/win6_gs.pdf



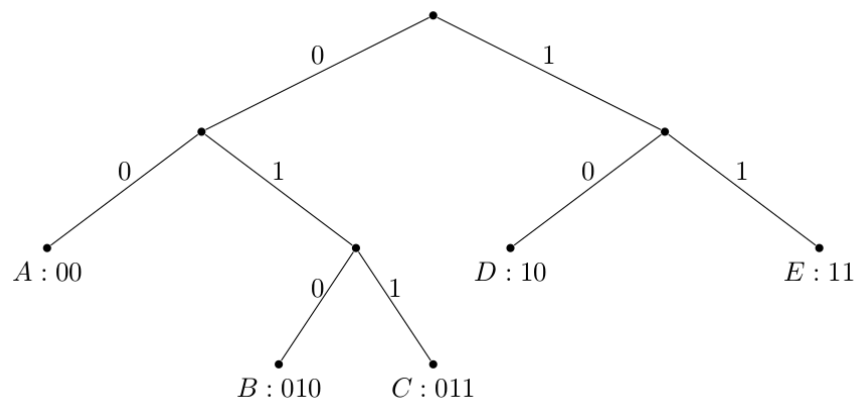
Fonte: https://www.researchgate.net/figure/Simplified-DEFLATE-algorithm_fig3_228411140

PNG (Predictor + Deflate)

- **Pré compressão** (Predictor - "Filter method")

A razão do uso do predictor ser eficiente deve-se ao facto de os dados terem uma correlação linear (Ou seja o valor seguinte tem uma grande relação de proximidade com o anterior como já referimos anteriormente). Acabamos assim por tirar vantagem disto utilizando um Filter method como por exemplo o Delta Filter explorando a redundância da imagem.

- **Compressão** (Algoritmo de Huffman de comprimentos fixos ou adaptativos)



Fonte: <https://leimao.github.io/blog/Huffman-Coding/>



Fonte: https://en.wikipedia.org/wiki/Portable_Network_Graphics#Tool_list

TÓPICOS

① Introdução

② Etapas de codificação





③ Algoritmos usados

④ Combinações mais usuais

⑤ Conclusão

Data set: Silesia Corpus

TABLE I. DESCRIPTION OF FILES ON SILESIA CORPUS



File Name	Data Type	Size (Bytes)	Description
dickens	Text in English	10,192,446	Collected works of Charles Dickens (from Project Gutenberg)
mozilla	Executable file	51,220,480	Tarred executables of Mozilla 1.0 (Tru64 Unix edition) (from Mozilla Project)
mr	Image file	9,970,564	Medical magnetic resonance image
nci	Database	33,553,445	Chemical database of structures
ooffice	Executable file	6,152,192	A dynamic linked library from OpenOffice.org 1.01
osdb	Database	10,085,684	Sample database in MySQL format from Open Source Database Benchmark
reymont	PDF file	6,625,583	Text of the book Chłopi by Władysław Reymont
samba	Executable file	21,606,400	Tarred source code of Samba 2-2.3 (from Samba Project)
sao	Binary Database	7,251,944	The SAO star catalogue (from Astronomical Catalogues and Catalogue Formats)
webster	HTML file	41,458,703	The 1913 Webster Unabridged Dictionary (from Project Gutenberg)
xml	XML file	5,345,280	Collected XML files
x-ray	Image file	8,474,240	X-ray medical picture

Fonte: <https://www.semanticscholar.org/paper/Modern-lossless-compression-techniques%3A-Review%2C-and-Gupta-Bansal/0a9a2f974b9696b680ecd5e4a57758140739697f>

Resultados

Os algoritmos foram comparados utilizando um máquina com Processador Intel (R) i5-3230M (velocidade de clock de 2,60 GHz), 12 GB de RAM e executando o Windows 8.1 Professional.

TABLE II. PERFORMANCE OF DEFLATE ON SILESIA CORPUS

File Name	Original Size (Bytes)	Final Size (Bytes)	Comp. Ratio	Comp. Speed (MB/s)	Decomp. Speed (MB/s)
dickens	10,192,446	3,851,823	2.646	8.572	72.002
mozilla	51,220,480	18,994,142	2.697	4.958	115.207
mr	9,970,564	3,673,940	2.714	4.999	98.028
nci	33,553,445	2,987,533	11.231	10.832	195.116
ooffice	6,152,192	3,090,442	1.991	7.512	40.186
osdb	10,085,684	3,716,342	2.714	14.798	102.324
reymont	6,625,583	1,820,834	3.639	4.264	107.096
samba	21,606,400	5,408,272	3.995	13.885	142.107
sao	7,251,944	5,327,041	1.361	9.296	82.333
webster	41,458,703	12,061,624	3.437	9.229	34.866
xml	5,345,280	662,284	8.071	13.522	37.760
x-ray	8,474,240	6,037,713	1.404	14.911	69.074
Average	-	-	3.825	9.732	91.342

TABLE III. PERFORMANCE OF LZMA ON SILESIA CORPUS

File Name	Original Size (Bytes)	Final Size (Bytes)	Comp. Ratio	Comp. Speed (MB/s)	Decomp. Speed (MB/s)
dickens	10,192,446	2,830,389	3.601	1.185	60.374
mozilla	51,220,480	13,362,705	3.833	1.643	53.095
mr	9,970,564	2,750,927	3.624	1.677	48.762
nci	33,553,445	1,529,702	21.934	1.440	193.934
ooffice	6,152,192	2,426,876	2.535	1.983	30.088
osdb	10,085,684	2,841,980	3.548	1.555	49.837
reymont	6,625,583	1,314,746	5.039	1.216	57.969
samba	21,606,400	3,740,983	5.775	1.964	82.094
sao	7,251,944	4,423,369	1.639	2.076	23.848
webster	41,458,703	8,369,031	4.953	1.065	31.404
xml	5,345,280	438,203	12.198	2.231	36.940
x-ray	8,474,240	4,491,727	1.886	1.996	24.564
Average	-	-	5.00	1.668	57.513

TABLE V. PERFORMANCE OF PPMd ON SILESIA CORPUS

File Name	Original Size (Bytes)	Final Size (Bytes)	Comp. Ratio	Comp. Speed (MB/s)	Decomp. Speed (MB/s)
dickens	10,192,446	2,327,178	4.380	3.542	3.377
mozilla	51,220,480	15,682,535	3.266	3.389	3.086
mr	9,970,564	2,335,937	4.268	4.483	4.222
nci	33,553,445	1,834,358	18.292	22.889	21.447
ooffice	6,152,192	2,496,315	2.464	2.945	2.695
osdb	10,085,684	2,354,180	4.284	3.762	3.608
reymont	6,625,583	1,052,589	6.295	5.192	4.860
samba	21,606,400	3,720,363	5.808	6.338	5.765
sao	7,251,944	4,759,281	1.524	1.775	1.565
webster	41,458,703	6,666,916	6.219	4.854	4.437
xml	5,345,280	377,581	14.157	15.930	11.746
x-ray	8,474,240	3,874,128	2.187	2.388	2.163
Average	-	-	6.095	6.457	5.748

Fonte: <https://www.semanticscholar.org/paper/Modern-lossless-compression-techniques%3A-Review%2C-and-Gupta-Bansal/0a9a2f974b9696b680ecd5e4a57758140739697f>

TAXA DE COMPRESSÃO

VELOCIDADE DE COMPRESSÃO / DESCOMPRESSÃO

E

Algoritmo de compressão de dados	Taxa de compressão (média)	Velocidade de compressão (média)MB/s	Velocidade de descompressão (média) MB/s
Deflate (Huffman + LZ77)	3.825	9.732	91.342
LZMA (Deriva do LZ77)	5.88	1.669	57.742
PPMd	6.095	6.457	5.748

Fonte: Modern Lossless Compression Techniques: Review, Comparison and Analysis

Obrigado pela
vossa atenção!

