Algoritmos em strings: compressão de texto

R. Rossetti, L. Ferreira, H. L. Cardoso, F. Andrade

FEUP, MIEIC, CAL

Teoria da Informação

- O que é?
  - "É uma ferramenta matemática para determinar a quantidade mínima de dados para representar informação"
- Representação da Informação
  - Como é que se representa texto?
  - Como é que se representam imagens?
  - Como é que se representa som?
  - Técnicas simples de correção de erros?
  - Dispositivos de armazenamento de informação?

Teoria da Informação

- Por que comprimir?
  - Preencher o hiato entre procura e capacidade
  - Utilizadores têm procurado aplicações com *media* cada vez mais sofisticados (Web 2.0, Big Data, data streaming em tempo real, …)
  - Meios de transmissão e armazenamento são limitados

    Por exemplo:
    Livro de 800 páginas; cada página com 40 linhas; cada linha com 80 caracteres:
    800 * 40 * 80 (* 1 byte por carácter) = 2,44 MB

    Vídeo digital c/ "qualidade de TV digital" (aproximadamente):
    1 segundo ~ 216Mbits
    2 horas ~ 194GB = 42 DVDs (ou 304 CD-ROMs)!

  - "compressão vai se tornar redundante em breve, com as capacidades de armazenamento e transmissão a aumentarem" … (Será?!!!)

Técnicas de compressão

- Codificador fonte e descodificador destino
  - Em sistemas multimédia, a informação é frequentemente comprimida antes de ser armazenada ou transmitida

    Algoritmos de compressão: principal tarefa do codificador fonte

    Algoritmos de descompressão: principal tarefa do descodificador destino

  - Implementação dos algoritmos de compressão/descodificação

    Em Software: quando o tempo para compressão/descompressão não é crítico

    Em Hardware: quando a aplicação é dependente do tempo, ou seja, quando o tempo para compressão/descompressão é crítico

Representação de carateres

- ASCII: *American Standard Code for Information Interchange*
- Tradicionalmente utilizava-se 7bits para representar os diversos caracteres
  - 7bits → 128 combinações diferentes possíveis
  - Por exemplo: 'A' = $(1000001)_2$ = $(65)_{10}$
- Mais tarde, os 7bits foram estendidos a 8, permitindo assim representar 256 caracteres diferentes
  - Unicode (16 bits) → 65.536
  - ISO* (32 bits) → 4.294.967.296

**International Organization for Standardization*

Representação de carateres

- Tabela ASCII (7 bits)

Left Digit(s) \ Right Digit	ASCII 0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	□	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

Representação de carateres

- Unicode (16 bits)

Code (Hex)	Character	Source
0041	A	English (Latin)
042F	Я	Russian (Cyrillic)
OE09	ก	Thai
13EA	Ꮺ	Cherokee
211E	℞	Letterlike Symbols
21CC	⇌	Arrows
282F	⠯	Braille
345F	㑯	Chinese/Japanese/ Korean (Common)

Representação de textos

- A representação de texto é simplesmente uma sequência de carateres

"O	L	A"	
79	76	65	Código ASCII
1001111	1001100	1000001	

Técnicas de compressão de texto

- Apesar do espaço de armazenamento estar continuamente a aumentar, é desejável por vezes comprimir dados
  - Transmissão pela rede
  - Armazenamento de longa duração
  - Em geral… maior eficiência e aproveitamento de recursos
- Três métodos comuns de compressão de texto (sem perdas)
  - *Keyword encoding*
  - *Run-length encoding (RLE)*
  - *Huffman codes*
- Na prática aplica-se muitas vezes uma combinação de técnicas

Keyword encoding

- Substituir palavras muito comuns por caracteres especiais ou sequências especiais de caracteres
- As palavras são substituídas de acordo com uma tabela de frequências (ocorrências)

Chave	Significado
%	carro
$	acidente
&	senhor
#	do

Keyword encoding (exemplo)

- "No acidente estiveram envolvidos três carros. O carro do senhor António ficou destruído. O carro do senhor José não sofreu grandes danos no acidente. O carro do senhor Carlos... bom, depois do acidente, nem se pode chamar aquilo um carro!"
  → 241bytes

- "No $ estiveram envolvidos três carros. O % # & António ficou destruído. O % # & José não sofreu grandes danos no $. O % # & Carlos... bom, depois # $, nem se pode chamar aquilo um %!"
  → 185bytes (76% do original)

Run-length encoding (RLE)

- Tipicamente utilizado quando o mesmo padrão/letra surge muitas vezes seguidas numa sequência de dados;
- Não é comum em texto, mas em muitos outros tipos de dados (por exemplo: imagem, vídeo)
- Técnica utilizada em muitas aplicações comuns. Basicamente, uma sequência de caracteres que se repetem é substituída por:
  - um marcador especial (*)
  - o carácter em questão
  - número vezes que o carácter aparece

AAAAAAAAAA	→	*A10
AABBBBBBBBAMMKKKKKKKKKM	→	AA*B8AMM*K9M

Algoritmo de Huffman

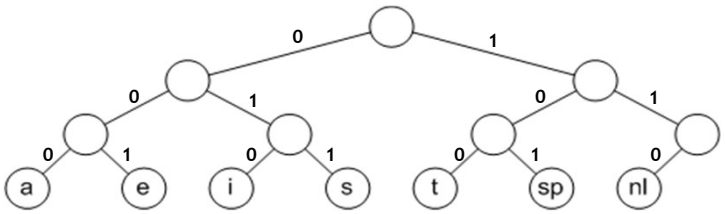
Codificação constante

- Código de tamanho fixo.
  - Se |alfabeto| = C → código com $\lceil \log_2(C) \rceil$ bits
  - Ex: caracteres ASCII visíveis ≅ 100 → necessário código de 7 bits
- Representação possível
  - Árvore binária com caracteres só nas folhas
  - Na descodificação:

    - se é folha, então encontrou-se o carácter
    - se o bit corrente do código for 0, visita-se a sub-árvore esquerda
    - se o bit corrente do código for 1, visita-se a sub-árvore direita

Codificação constante (exemplo)

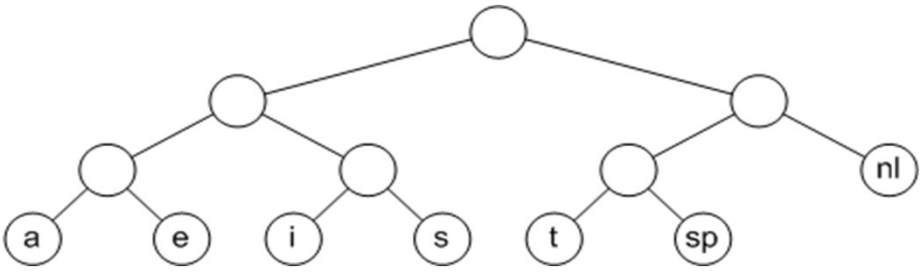
Carácter	Código	Frequência (f_i)	Total bits
a	000	10	30
e	001	15	45
i	010	12	36
s	011	3	9
t	100	4	12
espaço	101	13	39
newline	110	1	3
Total			



Custo da codificação $\Sigma f_i d_i$ = 174 bit

Codificação variável

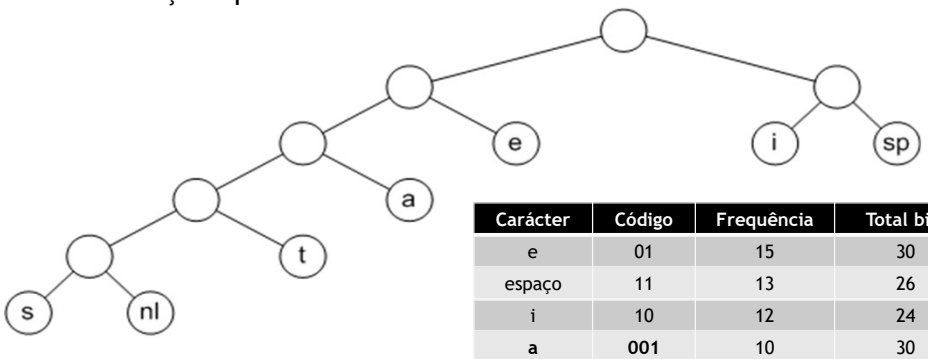
- Códigos de tamanho variável, para reduzir o custo de codificação …



Custo da codificação $\Sigma f_i d_i$ = 173 bit

Codificação variável (cont)

- Codificação óptima…



Carácter	Código	Frequência	Total bits
e	01	15	30
espaço	11	13	26
i	10	12	24
a	**001**	10	30
t	0001	4	16
s	00001	3	15
newline	00000	1	5
Total			**146**

Custo da codificação $\Sigma f_i d_i$ = 146 bit

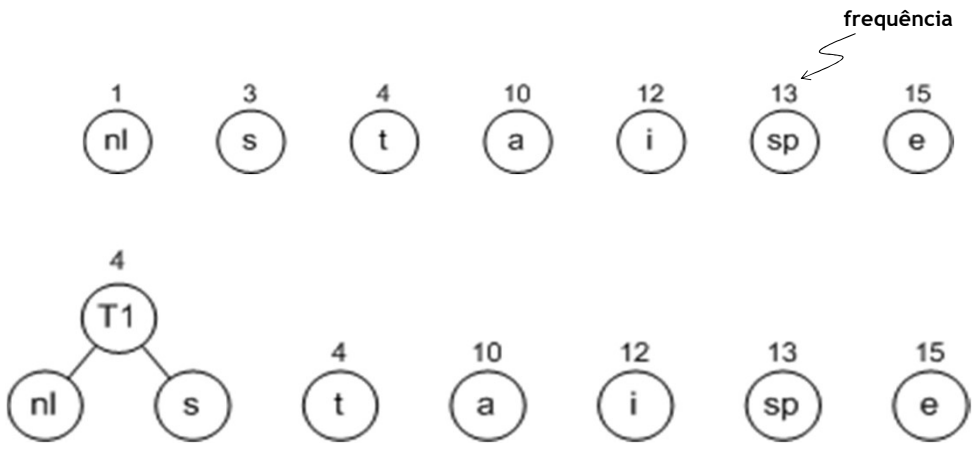
Códigos de Huffman

- Código de tamanho variável
  Caracteres mais frequentes → código mais pequeno
- Utiliza uma árvore binária com os símbolos só nas folhas
- Os símbolos nas folhas permitem descodificação não ambígua (código não prefixo)
- Ao usar uma árvore completa (*full tree*)
  todos os nós da árvore (excepto folhas) têm dois descendentes
- Minimiza o custo da codificação $\Sigma f_i d_i$
  onde f_i é a frequência relativa e d_i é a profundidade na árvore

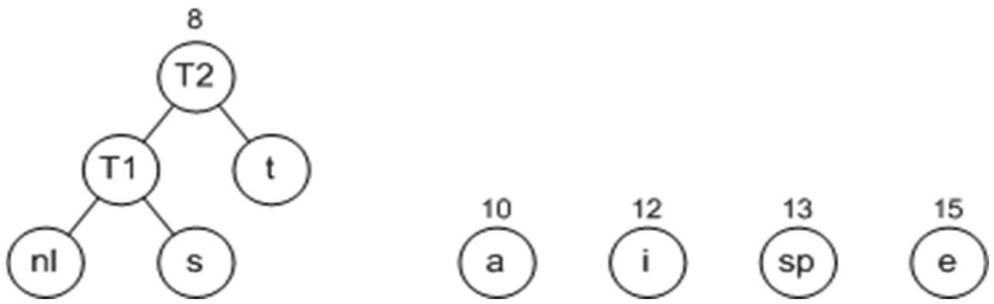
Algoritmo de Huffman

- O algoritmo de Huffman consiste de três passos básicos:
  - Cálculo da frequência de cada carácter no texto
  - Execução do algoritmo para construção de uma árvore binária
  - Codificação propriamente dita
- Inicialmente existe uma floresta de árvores só com raiz
- O peso de cada árvore é a soma das frequências relativas dos símbolos nas folhas
- Escolher as duas árvores com pesos menores e torná-las sub-árvores de uma nova raiz *(algoritmo ganancioso)*
- Repetir o passo anterior até haver uma só árvore
- Empates são resolvidos aleatoriamente

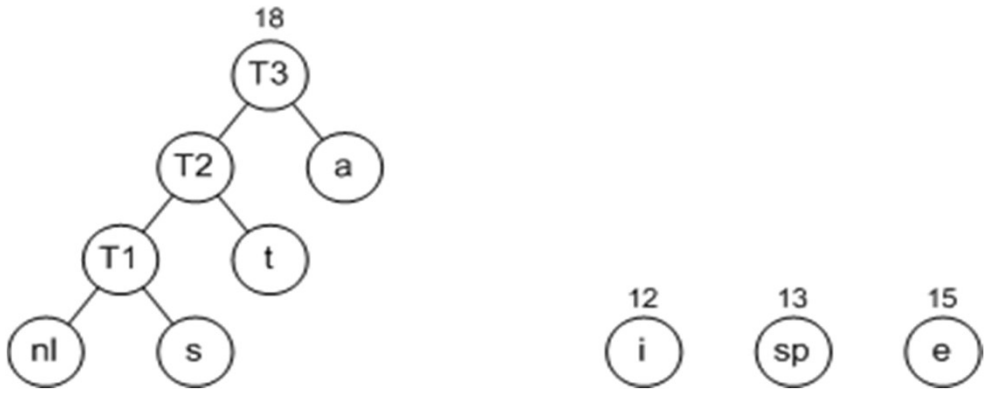
Algoritmo de Huffman (exemplo)



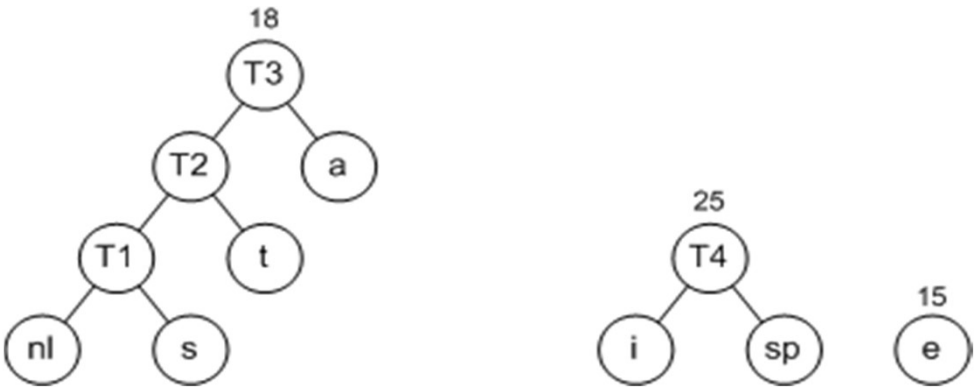
Algoritmo de Huffman (exemplo)



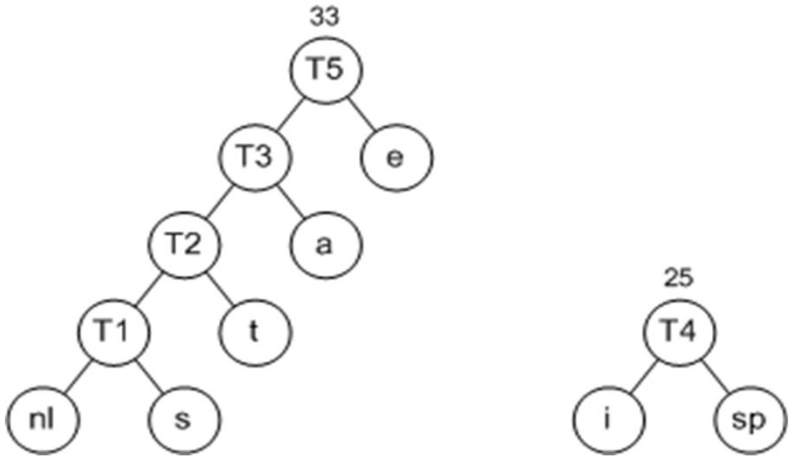
Algoritmo de Huffman (exemplo)



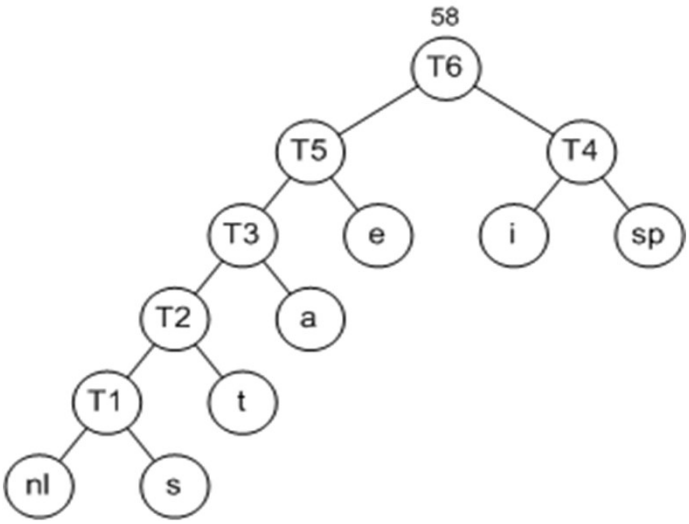
Algoritmo de Huffman (exemplo)



Algoritmo de Huffman (exemplo)



Algoritmo de Huffman (exemplo)



Algoritmo de Huffman

- Construção da árvore binária:

Algoritmo de Huffman:
Entrada: Um conjunto C de n caracteres.
Saída: Árvore de Huffman.

$n \leftarrow |C|$;
$Q \leftarrow C$;
para $i \leftarrow 1$ até $n-1$ faça
  $CriaNo(z)$;
  $x \leftarrow z.esq \leftarrow ExtraiMinimo(Q)$;
  $y \leftarrow z.dir \leftarrow ExtraiMinimo(Q)$;
  $f[z] \leftarrow f[x] + f[y]$;
  $Insere(Q, z)$;
retorne $ExtraiMinimo(Q)$;

Exercício: Compressão de texto (*Huffman*)

- Considere o texto "**pimpampumcadabolamataum**":
  - Defina um sistema de codificação constante para o texto acima. Qual é o tamanho mínimo do código e o custo de codificação para o texto dado?
  - Determine a árvore de codificação de Huffman para este texto, explicando detalhadamente todo o processo. Qual o custo de codificação neste caso?
    - https://people.ok.ubc.ca/ylucet/DS/Huffman.html
  - Utilizando a árvore de Huffman calculada na alínea anterior, apresente a codificação da frase "**pimpampum**" e o seu custo. Apresente também a codificação dos caracteres individualmente.

Referências e mais informação

- "Introduction to Algorithms", Second Edition, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press, 2001
- "The Algorithm Design Manual", Steven S. Skiena, Springer-Verlag, 1998
- David A. Huffman, *A Method for the Construction of Minimum-Redundancy Codes*, Proceedings of the Institute of Radio Engineers, 40(9):1098-1101
- Com base em slides de R. Camacho