Compressão de Sinais Multimedia

Compressão minimiza o tamanho dos dados, tirando partido de redundâncias estatísticas, temporais, espaciais, espectrais, estruturais e perceptívas.

- Critérios:
  - Taxa de compressão: $T_c = \frac{D_o}{D_c}$
  - Velocidade de processamento

- Compressão sem Perdas (Lossless)
  - Os dados originais são recuperados após a compressão
  - Taxa de compressão baixa
  - Usa-se em digitalização de documentos, obras de arte, medicina,etç
  - Ficheiros: zip, rar, …

- Compressão com Perdas (Lossy)
  - Há perda de informação
  - Perceptualmente esta perda é controlada
  - Taxa de compressão elevada
  - Usado em filmes, músicas, etç

Métodos de compressão sem perdas

- Run Length Coding (RLC)

- Variable Length Coding (VLC)
  - Algoritmo Shannon-Fano
  - Código Huffman
  - Código de Huffman adaptativo

- Codificação Aritmética

- Codificação Baseada em dicionário
  - LZ77
  - LZ78
  - Limpel Ziv Storer Szymanski (LZSS)
  - Limpel Ziv Welch (LZW)

- Codificação Diferencial, Codificação Preditiva

Run Length Coding (RLC)

- Método de compressão suportado nos formatos BMP e TIFF;
- Baseia-se na redundância de informação contida nos dados;

- RLC substitui uma sequência de valores repetidos (run) por dois valores:
  - Run count – conta o número de repetições
  - Run value – o valor repetido

- Exemplo:

  AAAAAAbbbCCCCCCd

  6A3b6C1d

- Repare-se que 16 símbolos são representados por apenas 8 códigos!
- No caso da letra “d” há uma expansão.

Run Length Coding (RLC)

- Aplicação a sinais unidimensionais, ex: supressão de troços silêncio

- A aplicação deste método em imagens a preto e branco (ex: fax ou página de um livro) atinge níveis de compressão mais elevados que em imagens a cores.

- Variantes do método:
  - Bit plane encoding: consiste em considerar uma imagem um conjunto de planos (um para cada bit)

- Também existem variantes do RLC, com perdas, que aumentam a taxa de compressão. Estas são usadas em imagens reais.

- Zero-level supression é outro método semelhante ao RLC.

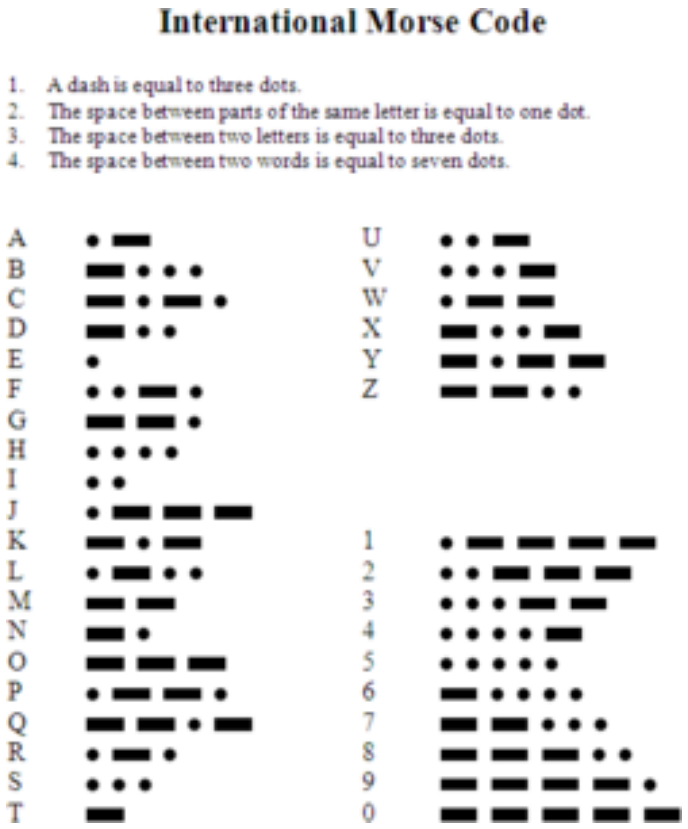
Codificação Entrópica de Sinais Multimedia

- Existe algum limite de compressão sem perdas óptimo que não possa ser melhorado?

  Esta questão é respondida pela Teoria da Informação.

- É de bom senso que valores que ocorram muitas vezes tenham um código mais pequeno e os que ocorrem menos vezes código maior.

- Esta noção tem por base o estudo estatístico dos dados (histograma)



Conceitos básicos de Teoria de Informação

- Considerando uma fonte de informação, sem memória, S cujo alfabeto é $S = \{s_1, s_2, \ldots, s_N\}$, onde os N símbolos são estatisticamente independentes,

- A informação própria é:

$$I(s_i) = \log_2 \left(\frac{1}{p(s_i)} \right) \quad \text{[bit]}$$

representa a quantidade de bits necessários para representar o símbolo s_i

- A entropia da fonte é:

$$\begin{aligned} H(S) &= \mathbf{E}[\log_2(1/p(s_i)] \\ &= -\sum_{i=1}^{N} p(s_i) \log_2 p(s_i) \quad \text{[bit/símbolo]} \end{aligned}$$

Mede a quantidade de informação codificada na mensagem.

- Nota: a entropia é tanto maior quando maior a incerteza.

Teorema da Codificação de fonte

- É possível codificar uma fonte, sem perdas, com um número médio de bits por símbolo maior ou igual à entropia da fonte:

$$H(S) < L < H(S) + \delta$$

- Codificação entrópica, consiste em codificar uma fonte por forma a atingir um número médio de bits por símbolo igual à entropia dessa fonte.
- Eficiência do código:

$$\eta = \frac{H(S)}{L}$$

onde, L é o número médio de bits por símbolo

$$\begin{aligned} L &= \mathbf{E}[L(s_i))] \\ &= \sum_{i=1}^{N} p(s_i) L(s_i) \end{aligned}$$

Conceitos básicos de Teoria de Informação

- A entropia está compreendida no intervalo:

$$0 \leq H(S) \leq \log_2(N)$$

- Exemplo: Imagem a preto e branco com $p_1 = p_2 = 1/2$

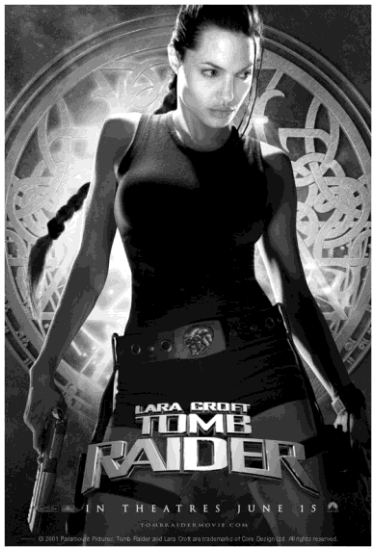
$$H(S) = ?$$

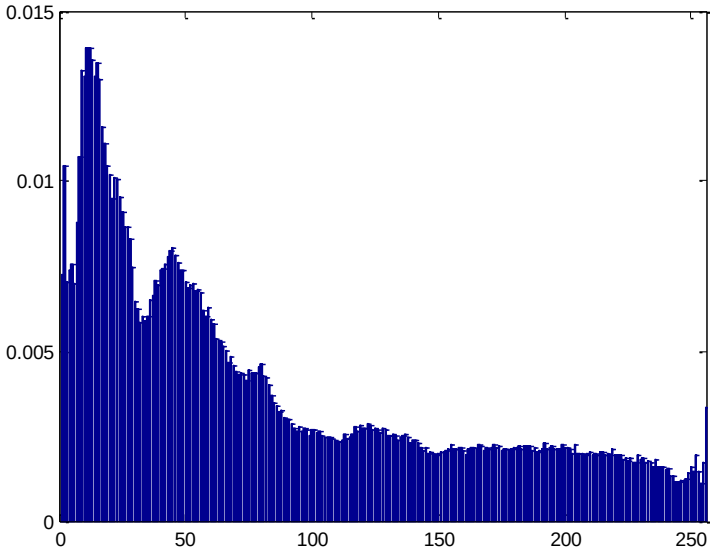
- Exemplo: fonte com 4 símbolos $p_1 = 5/8, p_2 = 1/8, p_3 = 1/8, p_4 = 1/8,$

$$H(S) = ?$$

- Exemplo: Imagem codificada com 256 niveis de cinzento

$$H(S) = ?$$





Algoritmo Shannon-Fano

- Método de codificação entrópica
- Código de comprimento variável (VLC)
- Método top-down

- Algoritmo
  - 1 – ordenar os símbolos por ordem decrescente de probabilidade (contabilizando o número de ocorrências de cada símbolo na mensagem)
  - 2 – Separar em dois grupos os símbolos com aproximadamente o mesmo número de occorrências
  - 3 – Atribuir a cada grupo o bit 0 e 1
  - 4 – voltar ao ponto dois até que cada grupo tenha apenas um símbolo

Algoritmo Shannon-Fano

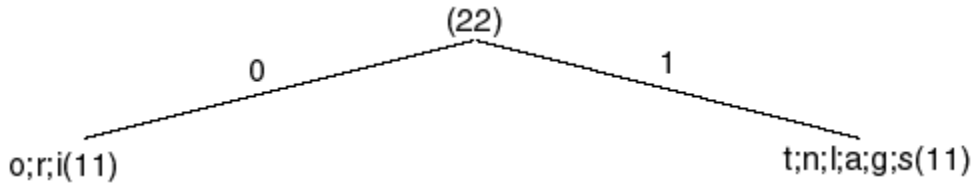
- Exemplo:
  - Mensagem: “otorrinolaringologista”
  - Símbolos / ocorrências

o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1

Algoritmo Shannon-Fano

- Exemplo:
  - Mensagem: “otorrinolaringologista”
  - Símbolos / ocorrências

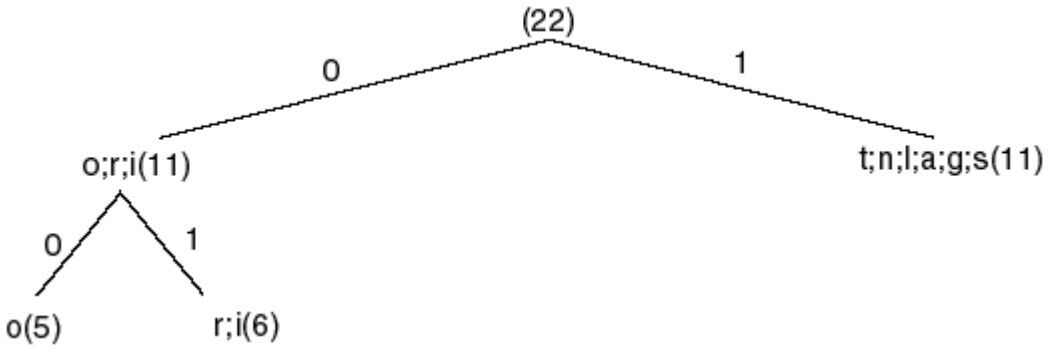
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
  - Mensagem: “otorrinolaringologista”
  - Símbolos / ocorrências

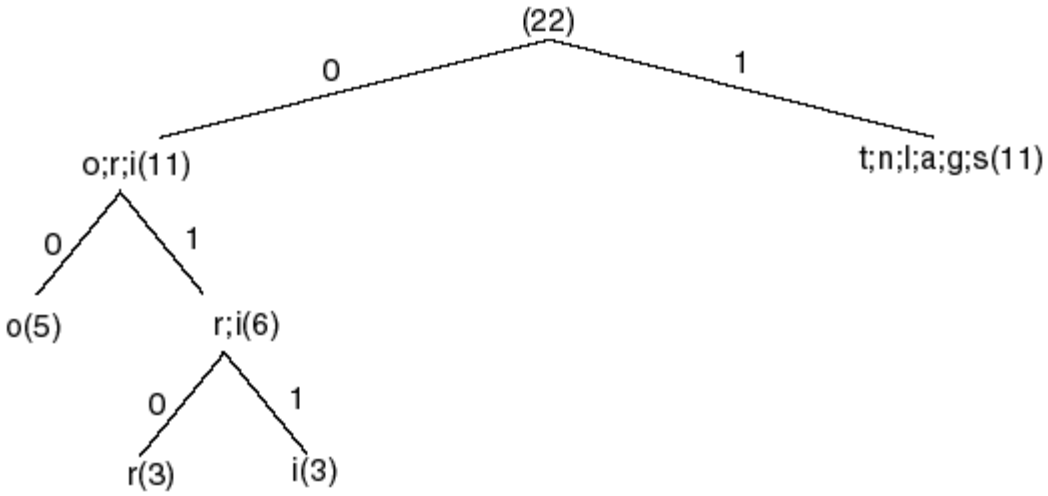
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
  - Mensagem: “otorrinolaringologista”
  - Símbolos / ocorrências

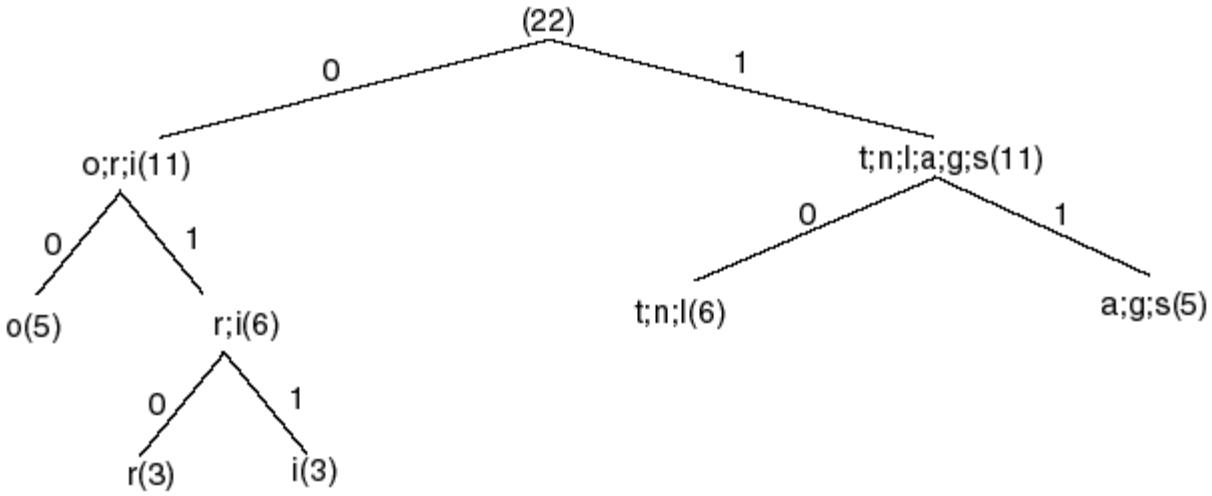
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
  - Mensagem: “otorrinolaringologista”
  - Símbolos / ocorrências

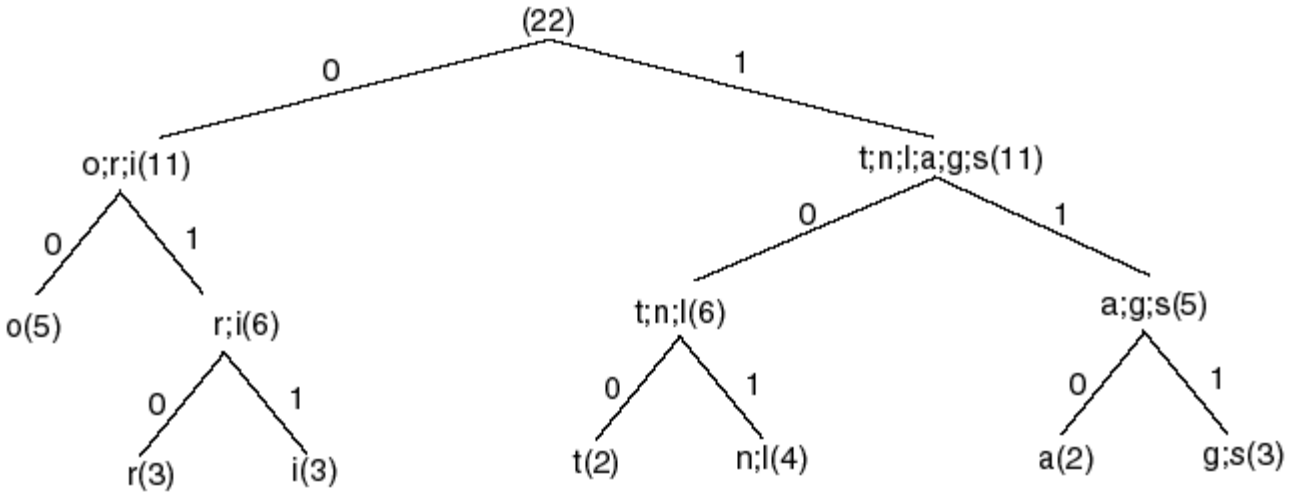
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
  - Mensagem: “otorrinolaringologista”
  - Símbolos / ocorrências

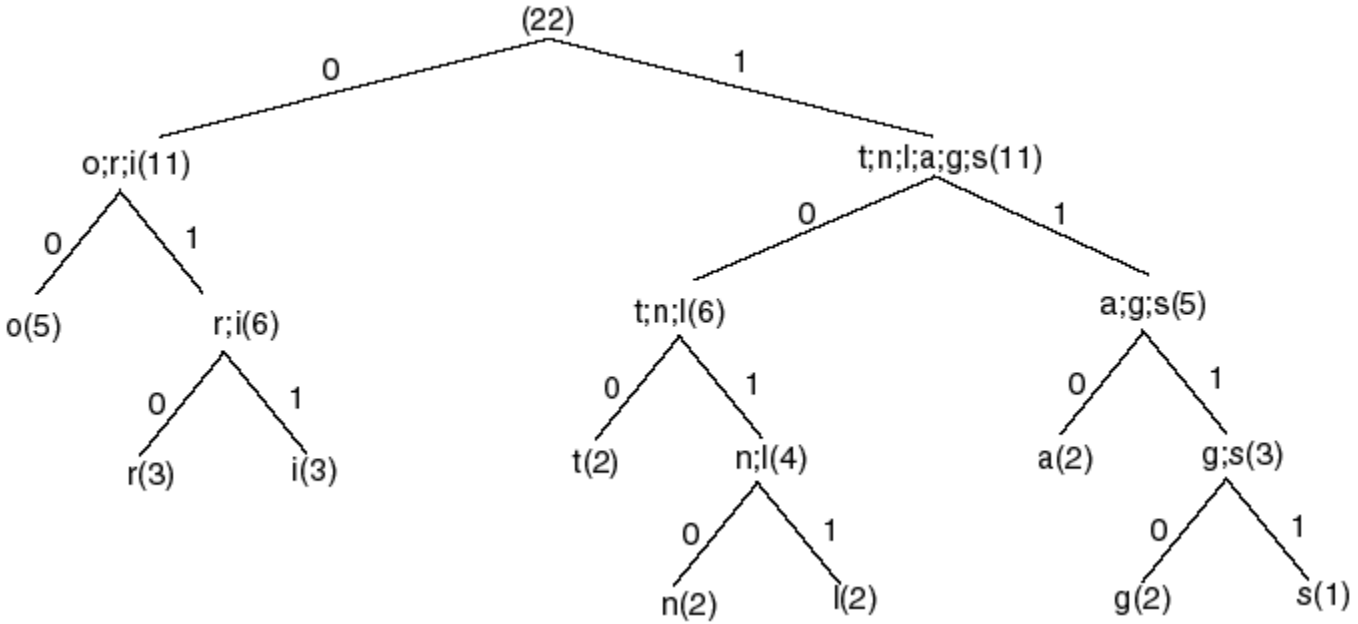
o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
  - Mensagem: “otorrinolaringologista”
  - Símbolos / ocorrências

o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- Exemplo:
  - Mensagem: “otorrinolaringologista”
  - Código de cada símbolo

o	5	00
r	3	010
i	3	011
t	2	100
n	2	1010
l	2	1011
a	2	110
g	2	1110
s	1	1111

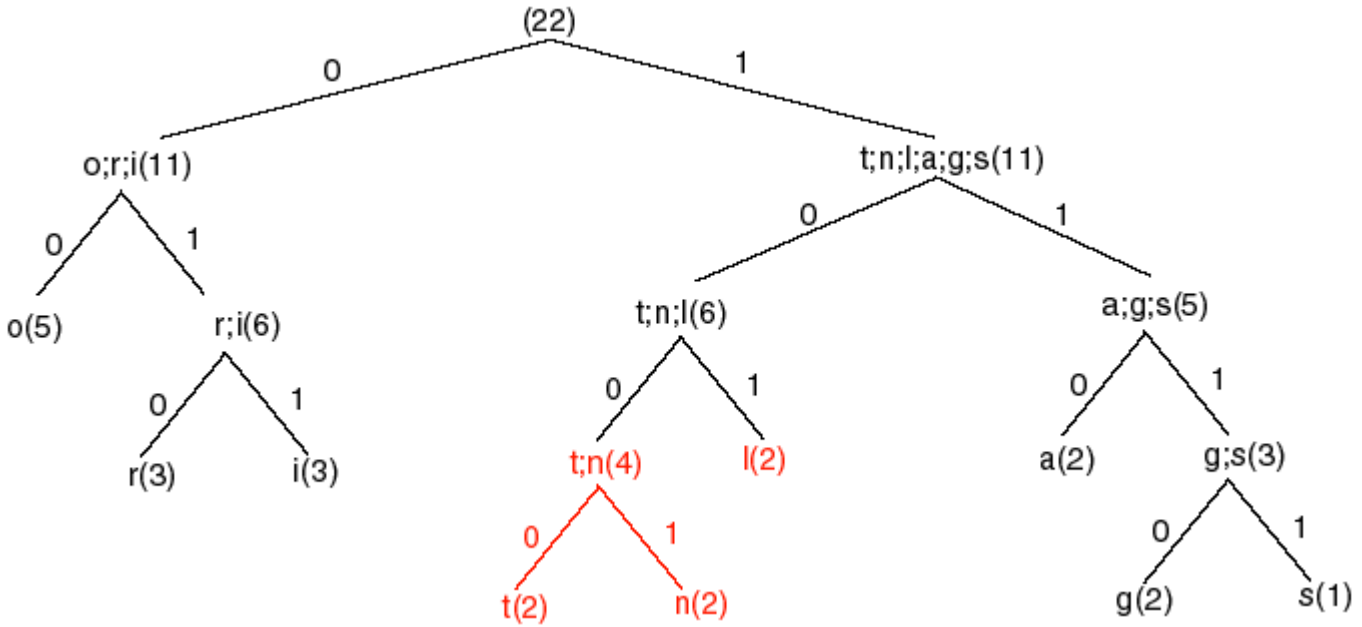
- Entropia = 3.045
- Número total de bits = 68
- Número médio de bits/ símbolo =3.091
- Codificação com código de tamanho constante:
  - 4 bits (9 símbolos)
  - Total = 88 bits

- Mensagem: “00 100 00 010 010 011 1010 00 1011 110 …”

Algoritmo Shannon-Fano

- Este algoritmo pode ter diferentes códigos para a mesma mensagem!
- Podendo alterar o número médio de bits por símbolo
- Exemplo:
  - Mensagem: “otorrinolaringologista”

o	r	i	t	n	l	a	g	s
5	3	3	2	2	2	2	2	1



Algoritmo Shannon-Fano

- O algritmo de descodificação é semelhante à descodificação do método de Huffman.

- Para melhorar o tempo de processamento existem outras alternativas ao invés de ler um bit de cada vez.

Algoritmo Huffman

- Método de codificação entrópica
- Código de comprimento variável (VLC)
- Método bottom-up
- Usado na compressão JPEG e MPEG

- Algoritmo
  - 1 – Colocar os símbolos numa lista por ordem decrescente de probabilidade
  - 2 – escolher os dois simbolos com menor frequência e agrupá-los num novo símbolo com probabilidade igual à soma destes dois.
  - 3 – Inserir o novo símbolo na lista ordenada.
  - 4 – voltar ao ponto 2 até que cada a lista tenha apenas um símbolo.
  - 5 – Atribuir códigos partindo do ultimo símbolo criado.

  - Nota: os códigos também se podem atribuir à medida que se cria o novo símbolo

Algoritmo Huffman

- Exemplo:
  - Mensagem: “otorrinolaringologista”

o (5)

r (3)

i (3)

t (2)

n (2)

l (2)

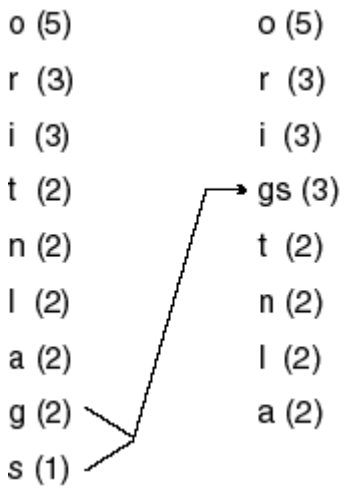
a (2)

g (2)

s (1)

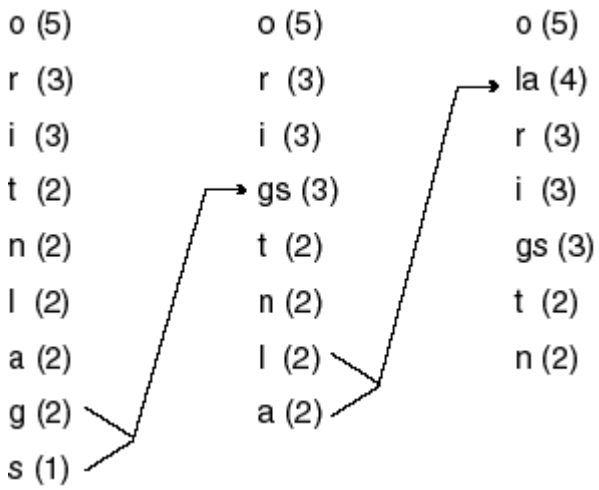
Algoritmo Huffman

- Exemplo:
  - Mensagem: “otorrinolaringologista”



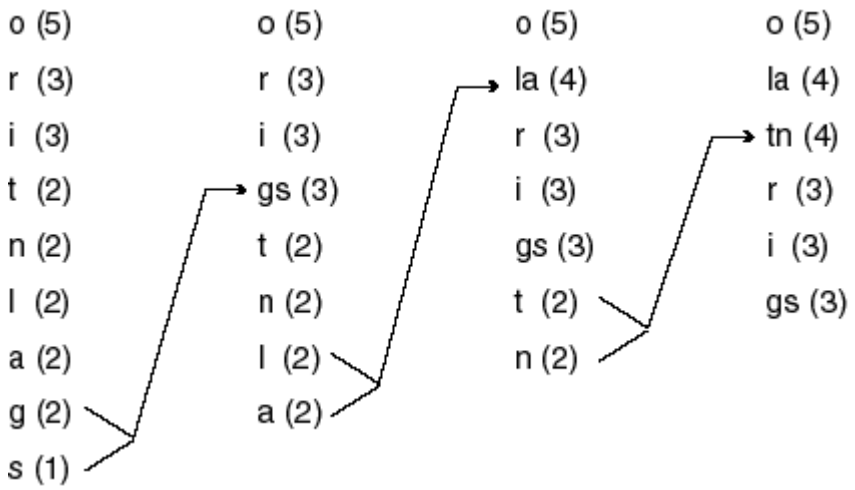
Algoritmo Huffman

- Exemplo:
  - Mensagem: “otorrinolaringologista”



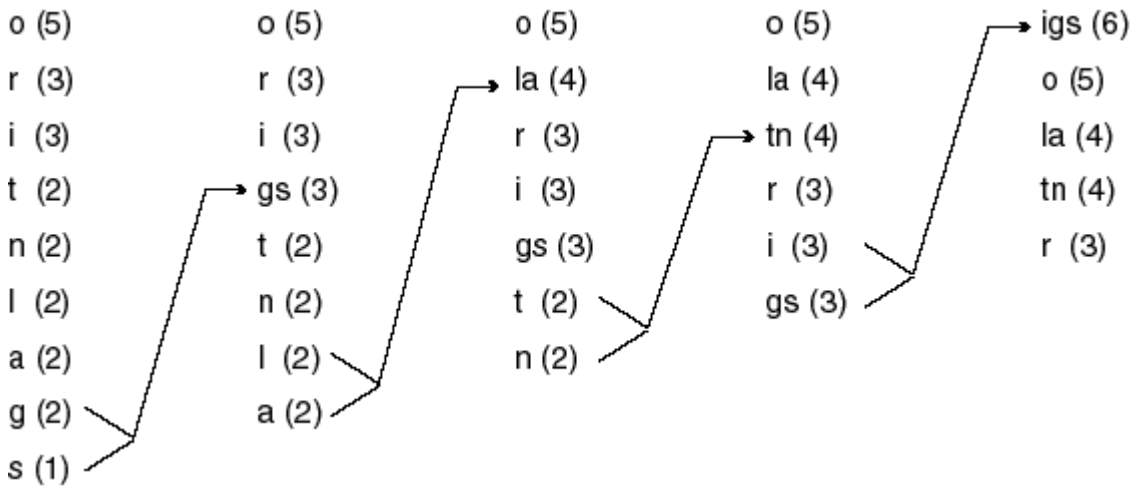
Algoritmo Huffman

- Exemplo:
  - Mensagem: “otorrinolaringologista”



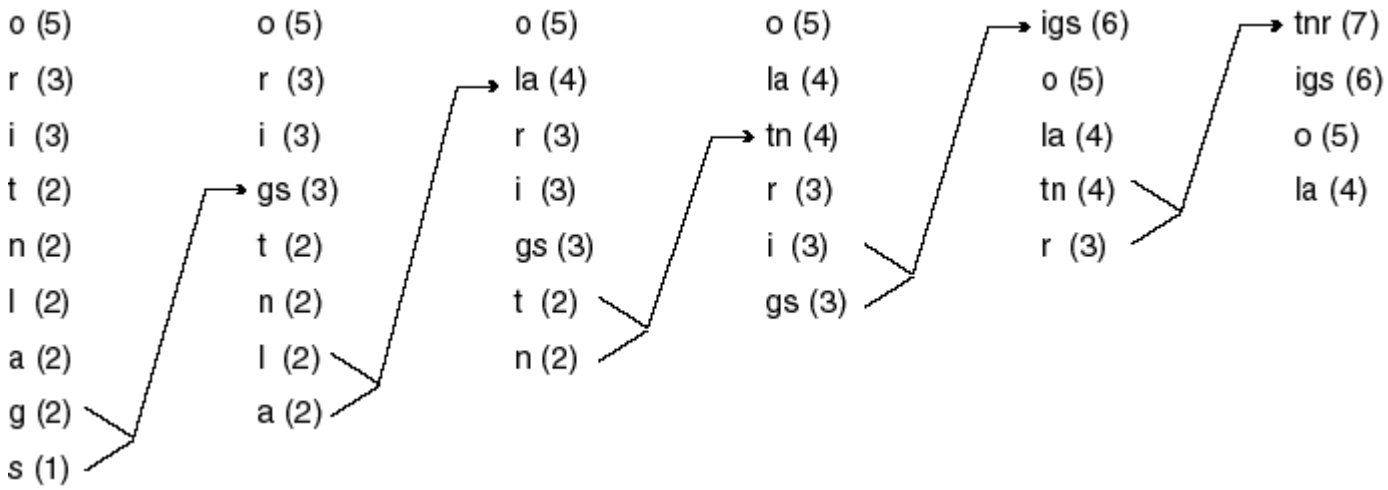
Algoritmo Huffman

- Exemplo:
  - Mensagem: “otorrinolaringologista”



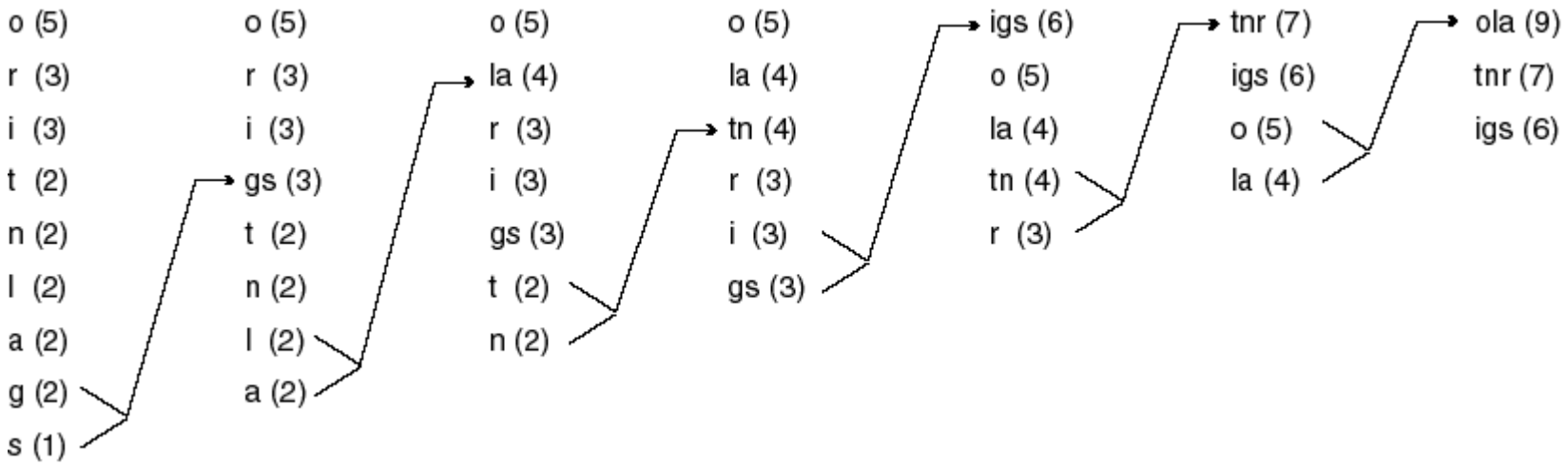
Algoritmo Huffman

- Exemplo:
  - Mensagem: “otorrinolaringologista”



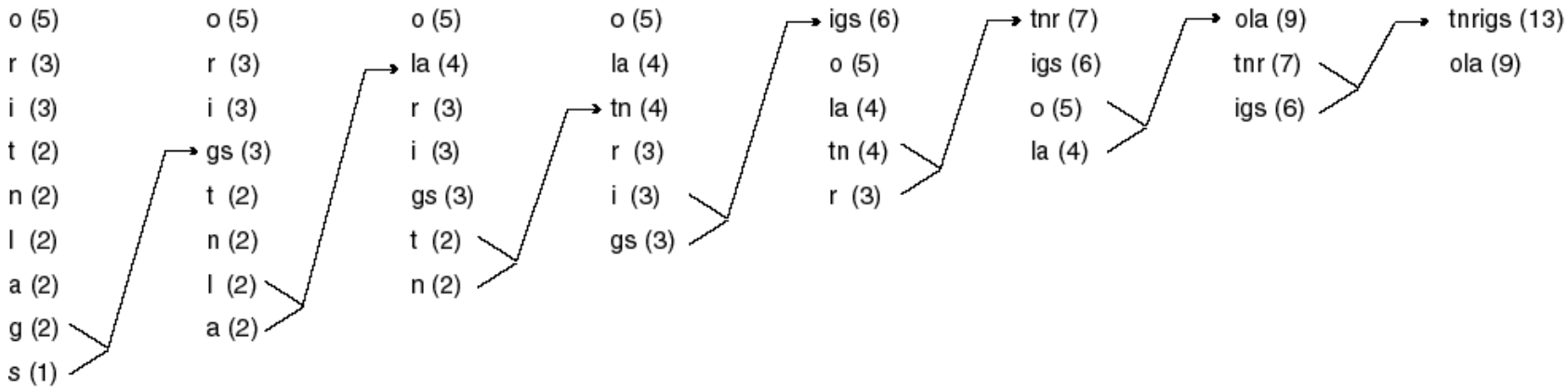
Algoritmo Huffman

- Exemplo:
  - Mensagem: “otorrinolaringologista”



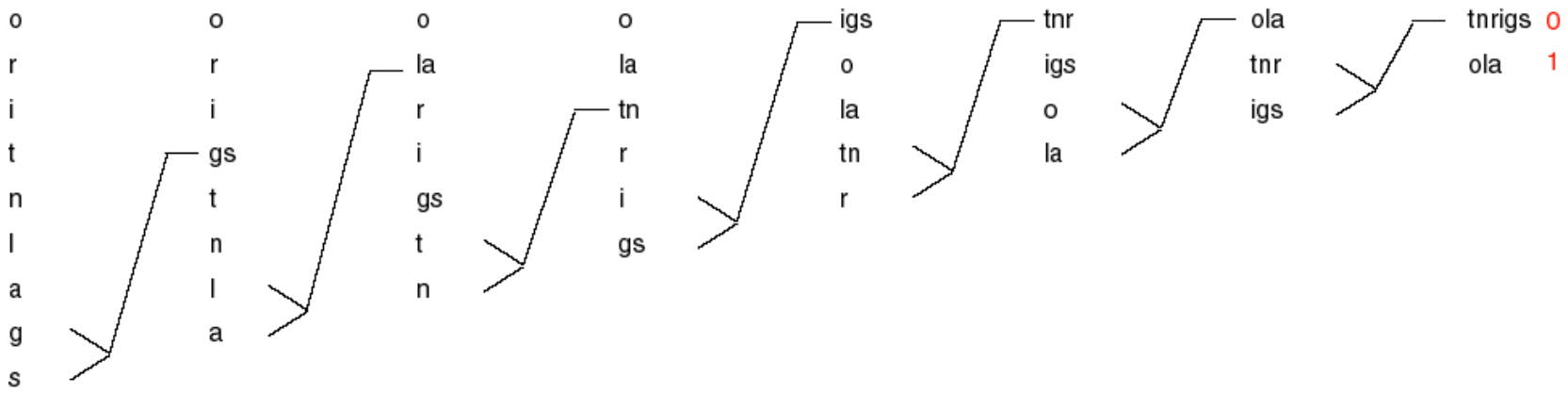
Algoritmo Huffman

- Exemplo:
  - Mensagem: “otorrinolaringologista”



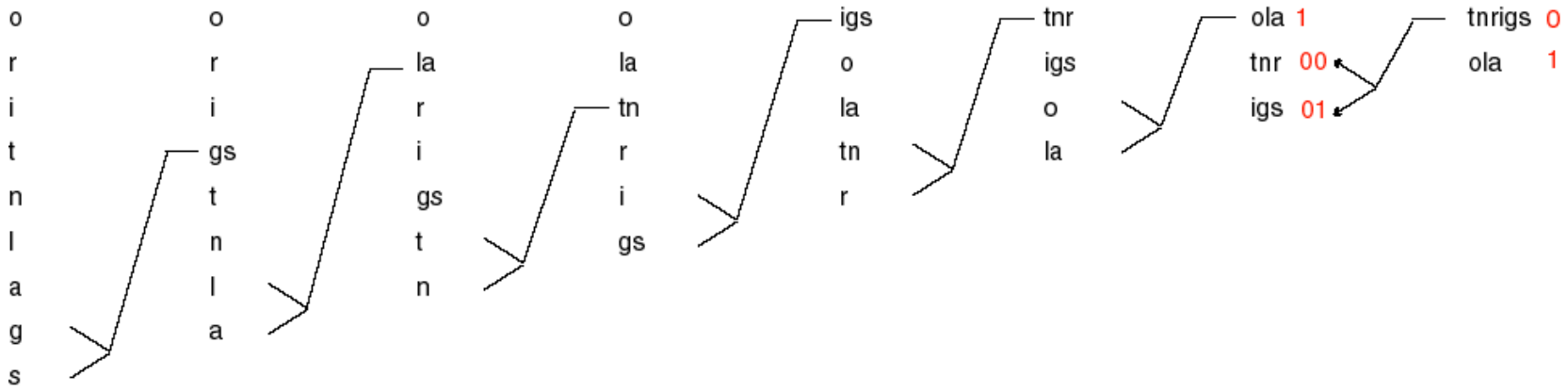
Algoritmo Huffman

- Exemplo:
  - Atribuição do código



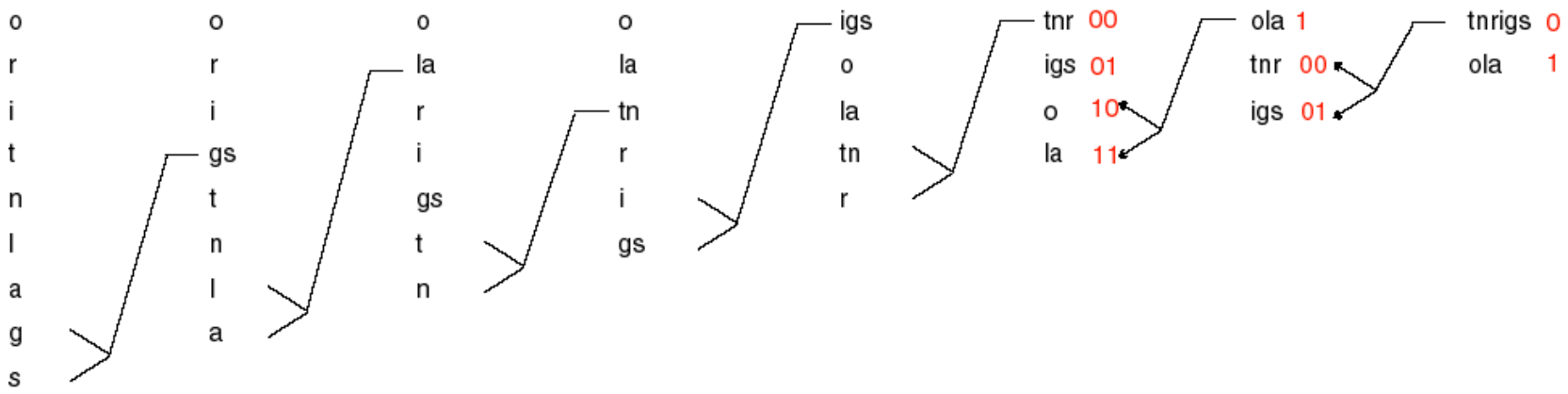
Algoritmo Huffman

- Exemplo:
  - Atribuição do código



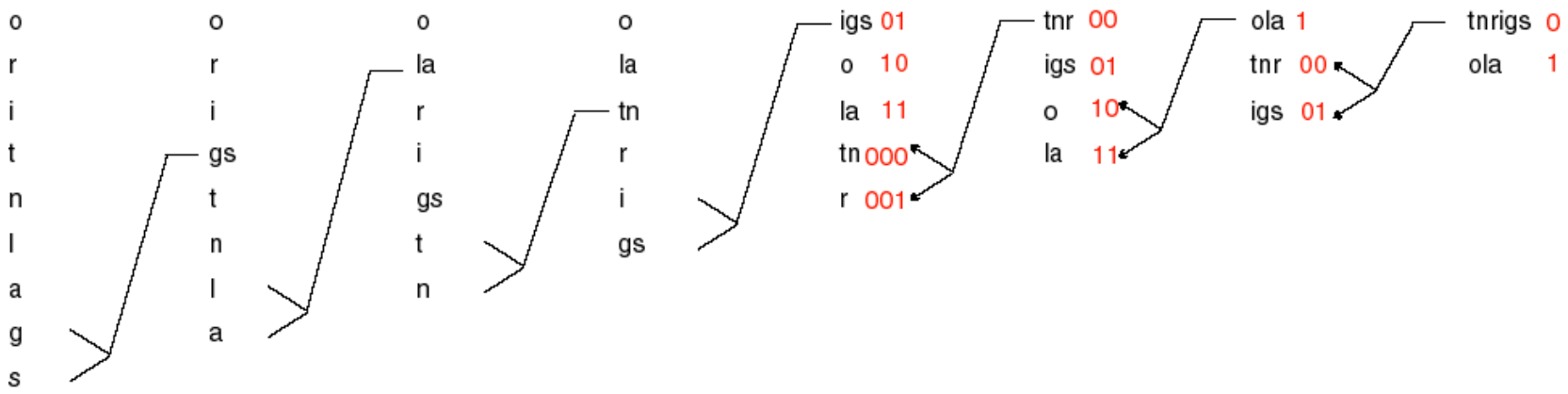
Algoritmo Huffman

- Exemplo:
  - Atribuição do código



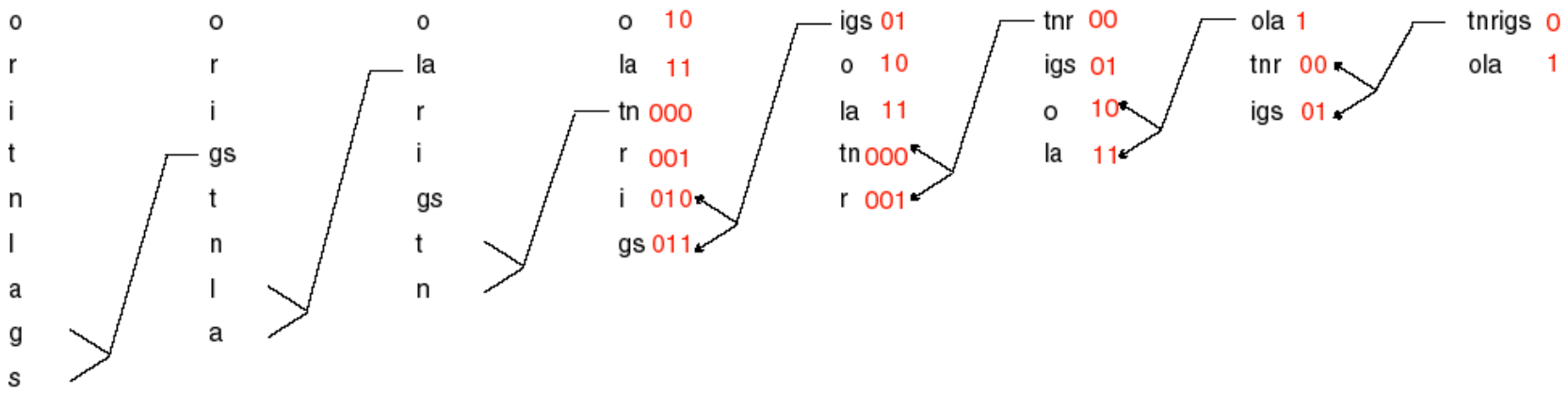
Algoritmo Huffman

- Exemplo:
  - Atribuição do código



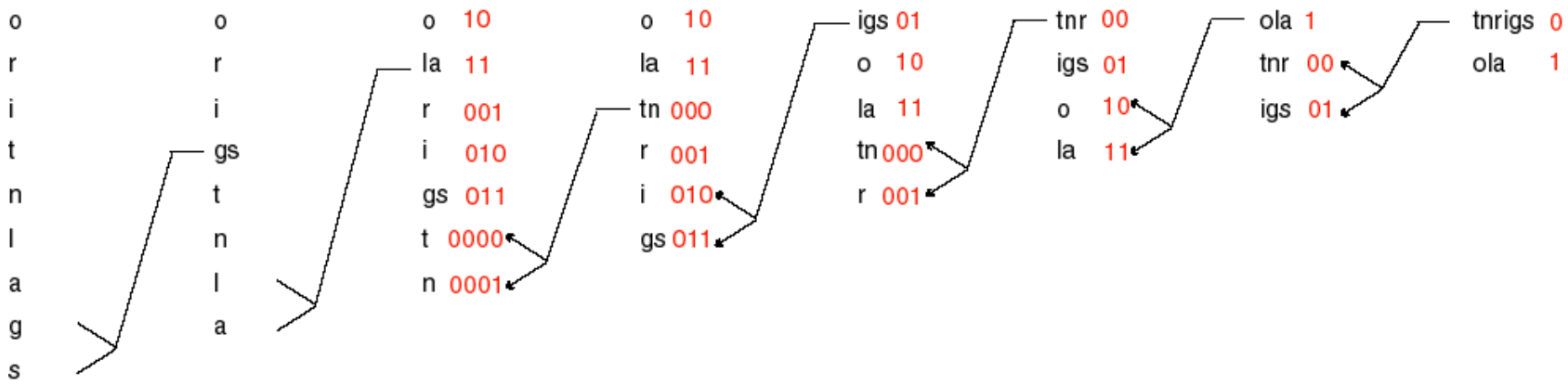
Algoritmo Huffman

- Exemplo:
  - Atribuição do código



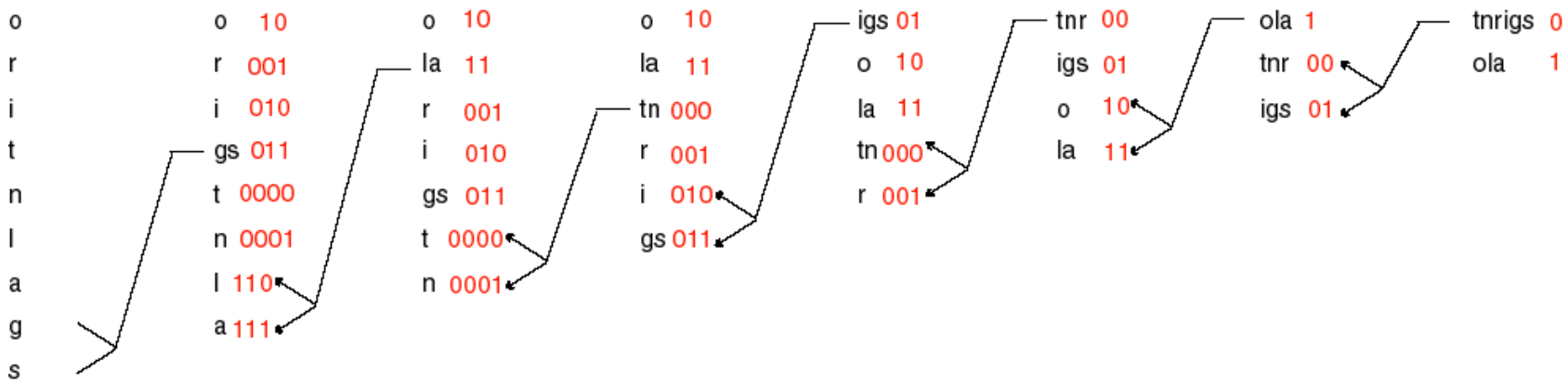
Algoritmo Huffman

- Exemplo:
  - Atribuição do código



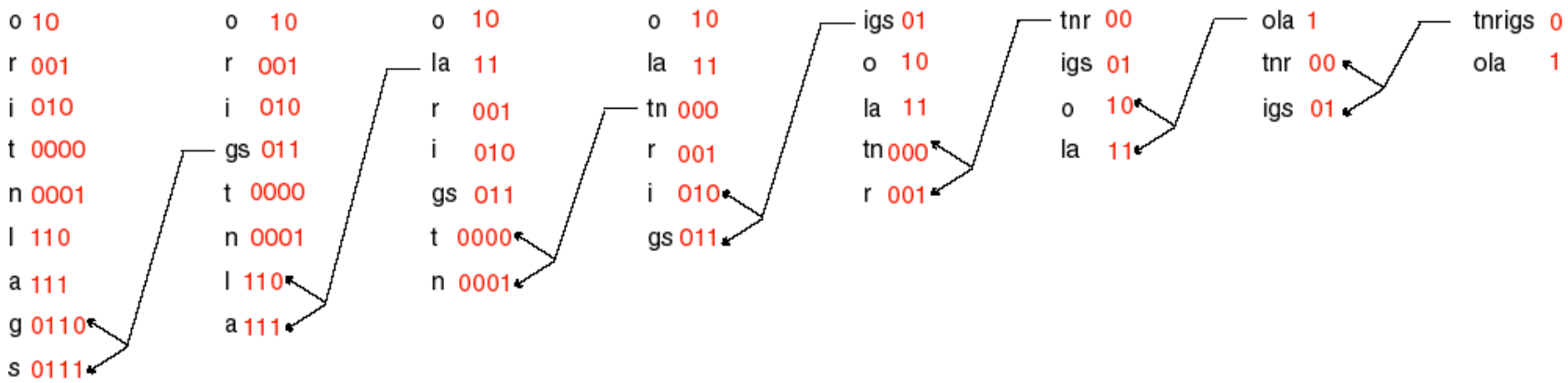
Algoritmo Huffman

- Exemplo:
  - Atribuição do código



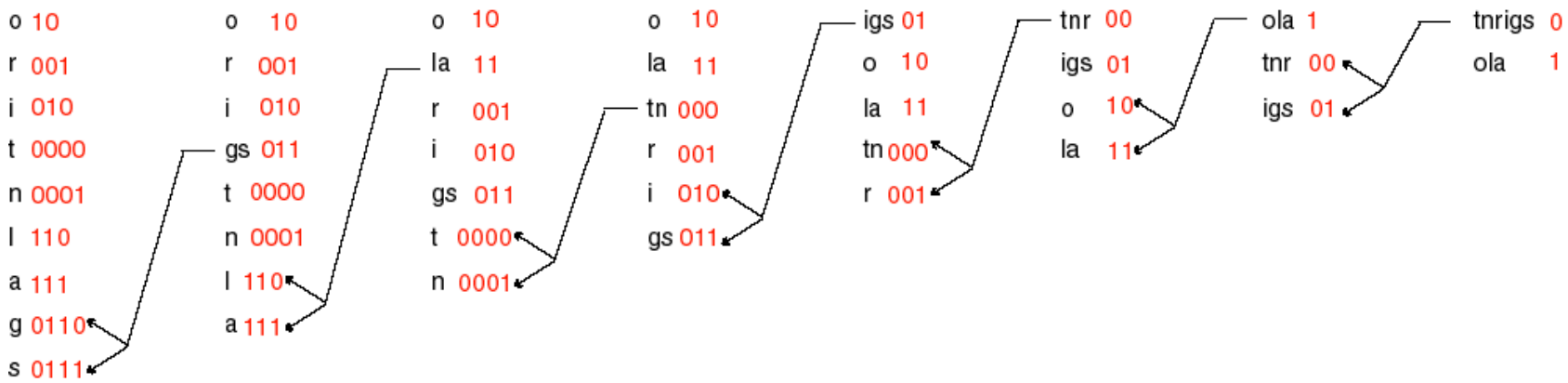
Algoritmo Huffman

- Exemplo:
  - Atribuição do código



Algoritmo Huffman

- Exemplo:
  - Atribuição do código



  - Mensagem: “10 0000 10 001 001 010 …”
  - Total de bits = 68

- O código de Huffman é óptimo dado que garante:

$$H(S) \leq L < H(S) + 1$$

- Propriedade: nenhum código é prefixo de outro!

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 10000010001001010 …
- Buffer:
- Output:

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 10000010001001010 …
- Buffer: 1
- Output:

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 1000001000100101010 …
- Buffer: 10
- Output:

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 10000010001001010 …
- Buffer:
- Output: o

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 1000001000100100101 0 …
- Buffer: 0
- Output: o

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 10000010001001010 …
- Buffer: 00
- Output: o

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 10000010001001010 …
- Buffer: 000
- Output: o

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 100000100010010 10 …
- Buffer: 0000
- Output: o

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 100000**10001001010** …
- Buffer:
- Output: ot

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 1000001**1**0001001010 …
- Buffer: 1
- Output: ot

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 1000001**0**001001010 …
- Buffer: 10
- Output: ot

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 10000010001001010 …
- Buffer:
- Output: oto

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 1000001000001001010 …
- Buffer: 0
- Output: oto

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 1000001000**0**1001010 …
- Buffer: 00
- Output: oto

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 1000001000**1**001010 …
- Buffer: 001
- Output: oto

Algoritmo Huffman

- Algoritmo de descodificação:
  - 1- ler bit para um buffer
  - 2- se o Código no buffer existe
    - Output simbolo e esvazia buffer
  - 3- Ir para o ponto 1

o	10
r	001
i	010
t	0000
n	0001
l	110
a	111
g	0110
s	0111

- Input: 100000100001001010 …
- Buffer: …
- Output: otor…

- Termina quando não houver mais bits

Algoritmo Huffman

- O código de Huffman:
  - Conhecimento à priori da densidade de probabilidade da fonte.
  - Esta informação tem de ser enviada no “header” do ficheiro

- No caso de sinais multimédia, muitas vezes não é possível obter este dado ou estima-lo com rigor.

- Código de Huffman adaptativo:

  - Inicialmente atribui-se um código a cada simbolo;
  - A estatistica é actualizada sempre que se recebe mais um símbolo;
  - A árvore é actualizada, mantendo as suas propriedades (troca de nós)
  - Na mesma mensagem o mesmo simbolo tem diferentes códigos;