Teoria da Informação

Charles Casimiro Cavalcante

charles@gtel.ufc.br

Grupo de Pesquisa em Telecomunicações Sem Fio – GTEL
Programa de Pós-Graduação em Engenharia de Teleinformática
Universidade Federal do Ceará – UFC
http://www.gtel.ufc.br/∼charles

"A principal função de um sistema de comunicação é reproduzir, exatamente ou de forma aproximada, uma informação proveniente de outro ponto diferente."

Claude Shannon, 1948

Conteúdo do curso

1. Revisão de probabilidade
2. Informação e Entropia
3. Codificação de fontes
4. Codificação e capacidade de canal
5. Complexidade de Kolmogorov
6. Funções de otimização
7. *Independent Component Analysis*

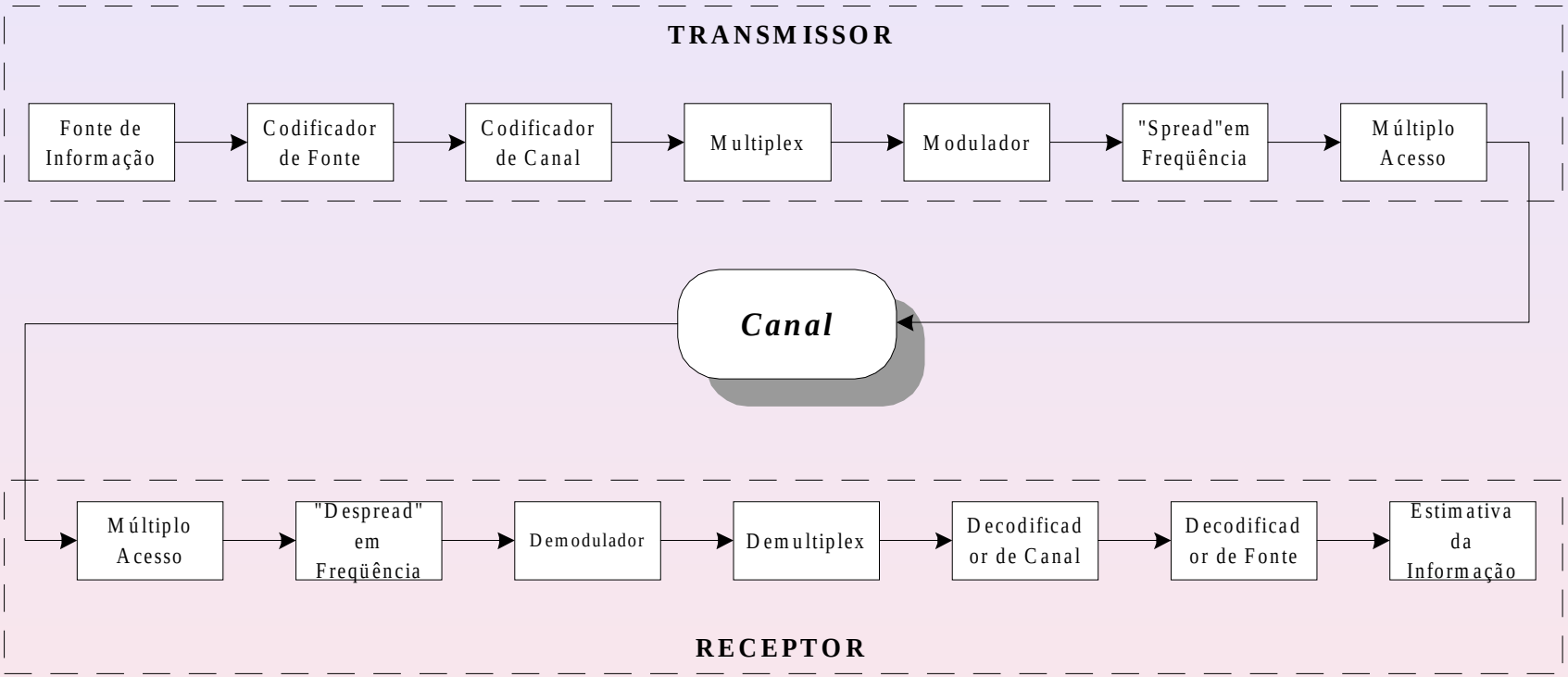
Parte III

Codificação de Fonte

Considerações

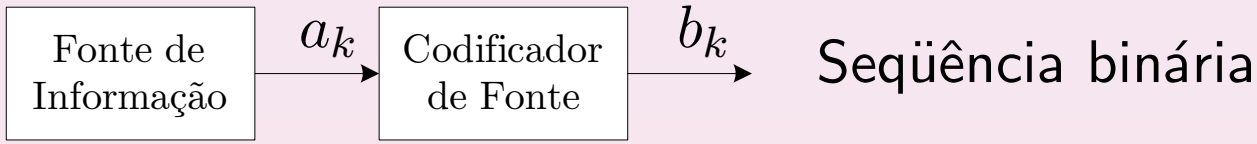
- Codificação de fonte é a representação dos dados da fonte de forma **eficiente**
- O dispositivo responsável por esta tarefa é chamado **codificador de fonte**
- Tipos de código
  - Símbolos mais freqüentes com *palavras* menores
  - Símbolos menos freqüentes (raros) com palavras maiores
- **Códigos de tamanho variável**
- Exemplo: código Morse (língua inglesa)
  - Letra mais freqüente: E ('.')
  - Letra menos freqüente: Q ('- - . -')

Sistema de comunicação - modelo geral



Codificador de fonte

- Palavras código são dadas na forma binária
- Código da fonte é *unicamente decodificável* tal que a seqüência de símbolos original pode ser reconstruída **perfeitamente** da seqüência binária codificada
- Subsistema:



Codificador de fonte - hipóteses (I)

- Seqüência da fonte de informação possui K diferentes símbolos
- Probabilidade de ocorrência do k-ésimo símbolo (a_k) é denominada p_k
- A palavra código (binária) associada ao símbolo a_k tem tamanho l_k
- Comprimento médio da palavra código: número médio de bits por símbolo da fonte usado na codificação

$$L = \sum_{k=0}^{K-1} p_k \cdot l_k \tag{76}$$

Codificador de fonte - hipóteses (II)

- Valor mínimo possível de L: L_{min}
- **Eficiência de codificação** do codificador de fonte

$$\eta = \frac{L_{\text{min}}}{L} \tag{77}$$

- Codificador é dito eficiente quando $\eta \to 1$
- **Como determinar o valor L_{min}?**

Primeiro teorema de Shannon: Teorema da codificação de fonte

- Também chamado de T*eorema da codificação sem ruído* - trata da condição de codificação sem erros
- Responde a questão fundamental da codificação de fonte

Teorema:

Dada uma fonte de informação discreta com entropia $\mathcal{H}(A)$, o tamanho médio da palavra código L para qualquer codificação de fonte sem distorção é limitado por

$$L \geq \mathcal{H}(A) \tag{78}$$

Codificação de fonte - limites

Entropia da fonte

Limite fundamental no número médio de bits para representar cada símbolo da fonte

Limite mínimo

$$L_{\text{min}} = \mathcal{H}(A) \tag{79}$$

Eficiência de codificação

$$\eta = \frac{\mathcal{H}(A)}{L} \tag{80}$$

O que fazer para achar o código?

- Remoção da redundância de informação do sinal a ser transmitido
- Processo geralmente chamado de *compactação de dados* ou *compressão sem perdas*
- Como gerar códigos com comprimento médio próximo da entropia?

Codificação prefixo

- Restrição de codificação de fonte: ser unicamente decodificável
- Nenhuma seqüência de palavras código correspondente a uma dada seqüência da fonte pode ser associada a outra seqüência qualquer

Código prefixo

É um código no qual nenhuma palavra código é **prefixo** de qualquer outra palavra código

Codificação prefixo - exemplos

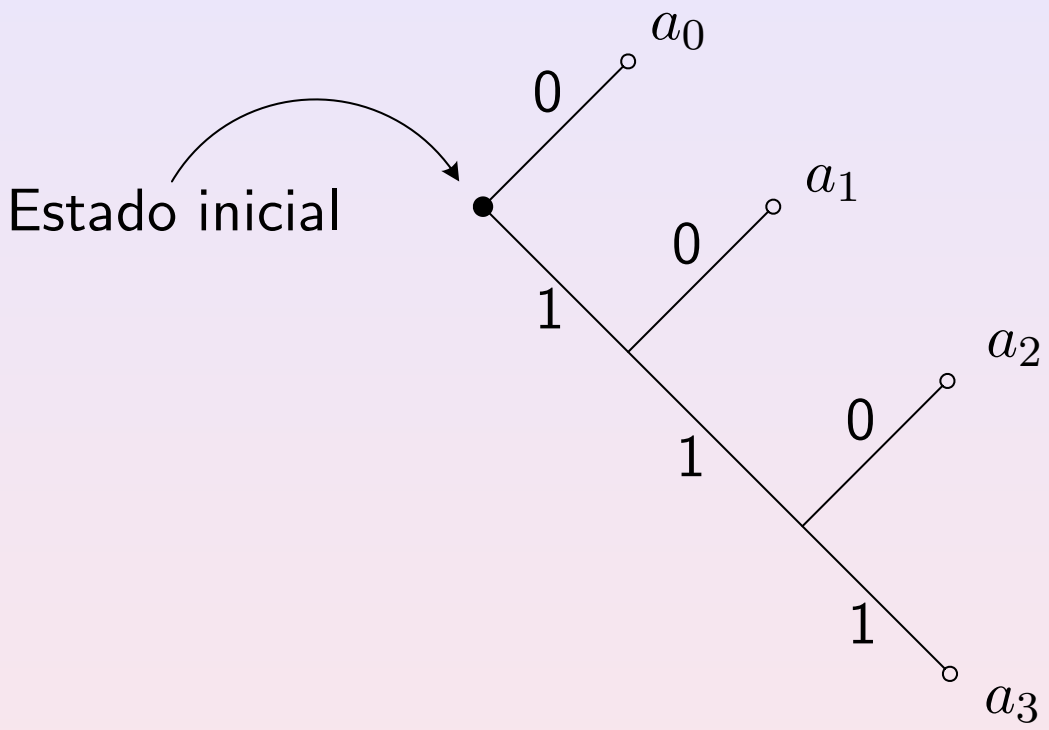
Símbolo da fonte	Probabilidade de ocorrência	**Código I**	**Código II**	**Código III**
a_0	0.5	0	0	0
a_1	0.25	1	10	01
a_2	0.125	00	110	011
a_3	0.125	11	111	0111

- **Código II é um código prefixo!**

Decodificação - código prefixo

- Busca numa *árvore de decisão*
- Facilidade de decodificação
- Cada bit que é recebido move o detector para algum ponto da árvore e um respectivo símbolo da fonte
- São **sempre** unicamente decodificáveis

Árvore de decisão - exemplo



- Decodificar 1011111000

Códigos prefixo

- Fonte discreta sem memória: $\{ a_0,\ a_1,\ \cdots, a_{K-1} \}$
- Probabilidades dos eventos: $\{ p_0,\ p_1,\ \cdots, p_{K-1} \}$
- Tamanho das palavras código: $\{ l_0,\ l_1,\ \cdots, l_{K-1} \}$

Inequação Kraft- McMillan

O tamanho das palavras códigos de um código prefixo devem satisfazer a seguinte inequação:

$$\sum_{k=0}^{K-1} 2^{-l_k} \leq 1 \tag{81}$$

Códigos prefixo - continuação

- Usando o inverso da definição da inequação de Kraft-McMillan pode-se dizer que

  Se os tamanhos das palavras código de um determinado código respeitam a inequação de Kraft-McMillan, então um código prefixo com aquelas palavras código pode ser construído

- Embora códigos prefixos sejam unicamente decodificáveis, o inverso não é verdade
- Outros códigos unicamente decodificáveis não permitem conhecer sempre o fim de uma palavra código
- Códigos prefixo são também chamados de **códigos instantâneos**

Códigos prefixo - equacionamento

- Seja uma fonte discreta sem memória com entropia $\mathcal{H}(A)$, o tamanho médio da palavra código de um código prefixo é limitado por

$$\mathcal{H}(A) \leq L < \mathcal{H}(A) + 1 \tag{82}$$

- O lado esquerdo de (82) é satisfeito com a igualdade sob a condição de

$$p_k = 2^{-l_k} \tag{83}$$

- Logo, pode-se escrever

$$\sum_{k=0}^{K-1} 2^{-l_k} \leq \sum_{k=0}^{K-1} p_k = 1 \tag{84}$$

Códigos prefixo - equacionamento (cont.)

- Usando a inequação de Kraft-McMillan (construção do código), o tamanho médio da palavra código é

$$L = \sum_{k=0}^{K-1} \frac{l_k}{2^{l_k}} \tag{85}$$

- Entropia da fonte

$$\begin{aligned} \mathcal{H}(A) &= -\sum_{k=0}^{K-1} \left(\frac{1}{2^{l_k}} \right) \log \left(2^{-l_k} \right) \\ &= \sum_{k=0}^{K-1} \frac{l_k}{2^{l_k}} \end{aligned} \tag{86}$$

- **Código prefixo apresenta a codificação mais eficiente neste caso**!

Códigos prefixo - equacionamento (cont.)

- Como encontrar o código prefixo para uma fonte arbitrária (probabilidades de ocorrência dos símbolos quaisquer valores)?
- Resposta: uso de **código estendido**.
- Seja L_n o tamanho médio da palavra código de um código prefixo estendido. Para um código unicamente decodificável L_n é o menor possível.
- Logo

$$\begin{aligned} \mathcal{H}(A^n) \leq L_n < \mathcal{H}(A^n) + 1 \\ n \cdot \mathcal{H}(A) \leq L_n < n \cdot \mathcal{H}(A) + 1 \\ \mathcal{H}(A) \leq \frac{L_n}{n} < \mathcal{H}(A) + \frac{1}{n} \end{aligned} \tag{87}$$

- No limite, os limites inferior e superior convergem

$$\lim_{n \to \infty} \frac{L_n}{n} = \mathcal{H}(A) \tag{88}$$

Códigos prefixo - equacionamento (cont.)

- Então, fazendo a ordem n de um código prefixo estendido grande o suficiente, o código pode representar, fidedignamente, uma fonte A tão próximo quanto se deseje
- Ou seja, o tamanho médio de uma palavra código de um código estendido pode ser tão pequeno quanto a entropia da fonte, dado que o código estendido tem um tamanho *suficiente*
- There is no free lunch!!!
- Complexidade de decodificação: aumenta na ordem do tamanho do código estendido

Codificação de Huffman

- Idéia básica: associar a cada símbolo, uma seqüência de bits, aproximadamente igual a quantidade de informação existente no símbolo
- Tamanho médio da palavra código aproxima o limite (entropia)
- Substituição do conjunto de estatísticas da fonte por um conjunto **mais simples**
- Processo de redução interativo até que o restem apenas dois símbolos para os quais '0' e '1' é um código ótimo
- Construção do código é feita através da retropropagação

Codificação de Huffman - algoritmo

1. Os símbolos da fonte são listados em ordem decrescente das probabilidades. Os dois símbolos com as menores probabilidades são assinalados como 0 e 1
2. Os dois símbolos anteriormente assinalados são *combinados* em um novo símbolo com a soma das probabilidades dos dois primeiros. A lista decrescente de probabilidades é novamente feita.
3. Procedimento se repete até sobrarem somente dois símbolos
4. Do final para o início da tabela, a palavra código de cada símbolo é encontrada

Codificação de Huffman - exemplo

- Seja a fonte discreta com as seguintes probabilidades para seus símbolos

Símbolo	Probabilidade
a_0	0.4
a_1	0.2
a_2	0.2
a_3	0.1
a_4	0.1

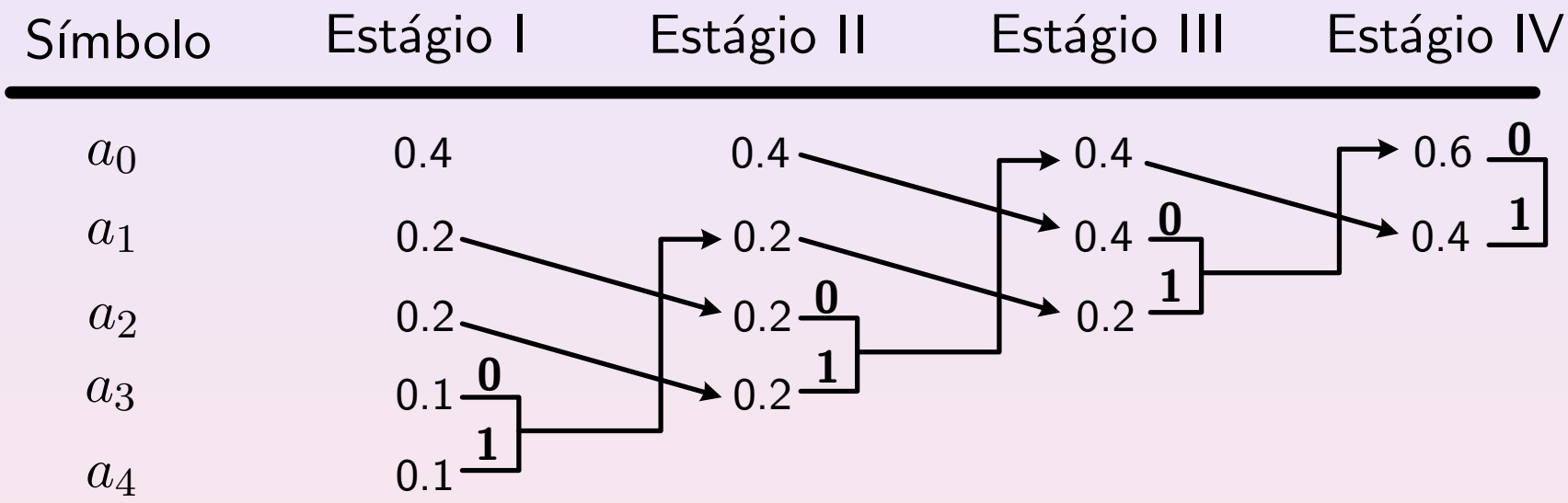
- Entropia da fonte

$$\mathcal{H}(A) = 2.12193$$

Pergunta

O código de Huffman vai fornecer um tamanho médio da palavra código próximo da entropia da fonte?

Codificação de Huffman - exemplo



Codificação de Huffman - exemplo

Símbolo	Probabilidade	Palavra código
a_0	0.4	00
a_1	0.2	10
a_2	0.2	11
a_3	0.1	010
a_4	0.1	011

- Tamanho médio palavra código

$$L = 2.2$$

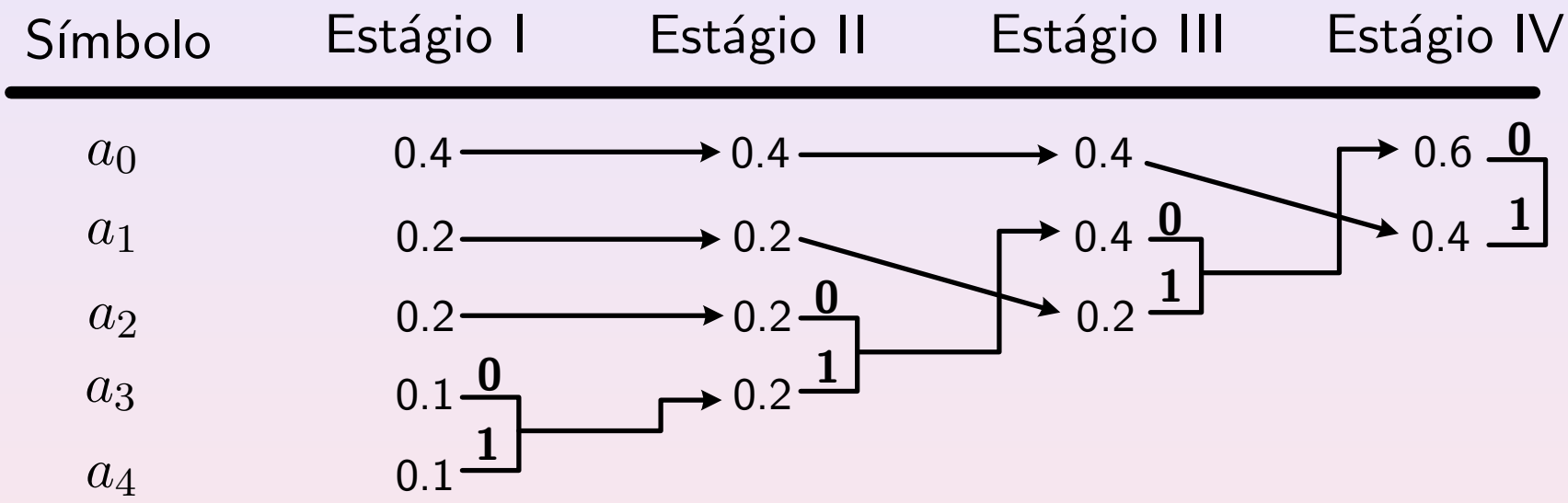
- L excede $\mathcal{H}(A)$ de apenas 3.67%
- L satisfaz $\mathcal{H}(A) \leq L < \mathcal{H}(A) + 1$

Codificação de Huffman - comentários

- A codificação de Huffman não é única!
- Maneiras
  1. Arbitrariedade da associação de 0 e 1
  2. Valor da probabilidade igual: colocação dos valores
  3. Diferentes tamanhos médios da palavra código
- Medida da variabilidade (variância) do tamanho médio das palavras código

$$\sigma^2 = \sum_{k=0}^{K-1} p_k \left(l_k - L\right)^2 \tag{89}$$

Codificação de Huffman - não unicidade



Codificação de Huffman - não unicidade

Símbolo	Probabilidade	Palavra código
a_0	0.4	1
a_1	0.2	01
a_2	0.2	000
a_3	0.1	0010
a_4	0.1	0011

$$\sigma_1^2 = 0.16$$

$$\sigma_2^2 = 1.36$$

Codificação de Huffman - problemas

1. Requer conhecimento *a priori* do modelo probabilístico da fonte
2. Na prática, são poucas as situações que se conhece *a priori* as estatísticas da fonte
3. Além disso, em algumas aplicações, como modelamento de texto, a armazenagem impede o código de Huffman de capturar as relações entre palavras e frase - comprometimento da eficiência do código
4. **Alternativa??**

Codificação de Lempel-Ziv

- Codificação adaptativa por natureza
- Implementação mais simples que a de Huffman

Idéia básica

Organização da seqüência de dados da fonte em segmentos que são subseqüências menores não encontradas anteriormente

Codificação de Lempel-Ziv - exemplo

- Seqüência de dados: **000101110010100101**
- Assume-se que os símbolos '0' e '1' já estão armazenados
- Logo, o início do algoritmo:
  Subseqüências armazenadas: 0, 1
  Dados para organização: 000101110010100101
- Menor seqüência **não** armazenada?
  Subseqüências armazenadas: 0, 1, 00
  Dados para organização: **0101110010100101**
- Outra seqüência?
  Subseqüências armazenadas: 0, 1, 00, 01
  Dados para organização: **01110010100101**
- Mais uma?
  Subseqüências armazenadas: 0, 1, 00, 01, 011
  Dados para organização: **10010100101**

Codificação de Lempel-Ziv - exemplo

Tabela de codificação

Pos.:	1	2	3	4	5	6	7	8	9
Subseq.:	**0**	**1**	**00**	**01**	**011**	**10**	**010**	**100**	**101**
Rep. num.:			11	12	42	21	41	61	62
Blocos cod.:			0010	0011	1001	0100	1000	1100	1101

Codificação de Lempel-Ziv - palavra código

1. Contagem do número de seqüências diferentes
2. Cálculo do número de bits n_1 para representar as seqüências
3. Número de bits da palavra código: $n_1 + 1$
4. Identificação da inovação e do prefixo
5. Identificação da **posição** do prefixo na tabela
6. Palavra código: **prefixo + inovação**

Codificação de Lempel-Ziv - características

- O último símbolo de cada subseqüência é chamado de **símbolo de inovação** - acrescenta à informação anterior e torna uma seqüência distinta
- O último bit de cada seqüência codificada representa o símbolo de inovação da subseqüência considerada
- Os bits restantes fornecem a representação equivalente do **ponteiro** para as *subseqüências originais*
- Decodificação: utiliza o ponteiro para identificar a subseqüência e adiciona o símbolo de inovação

Codificação de Lempel-Ziv - observações

- Código utiliza um número fixo de bits para representar um número variável de símbolos da fonte
- Na prática, 12 bits são utilizados, implicando num código de 4096 entradas
- Lempel-Ziv é o padrão de compactação de dados
- Para textos, Lempel-Ziv atinge uma compactação de aproximadamente 55% contra 43% de Huffman

Codificação de Lempel-Ziv - exemplos

- http://www.data-compression.com/lempelziv.html

- http://www.data-compression.com/theory.shtml

3.1 – LZ77

- Cada seqüência de caracteres é codificado como a trinca $< o, l, c >$, onde
  - o indica quantos caracteres para trás da posição atual do buffer está o início do casamento;
  - l indica o tamanho do casamento;
  - c indica um caracter de inovação que é adicionado depois que acaba o casamento.
- Exemplo 1:

$$\ldots cabracadabrarrarrad \ldots$$

  - Supondo que a janela total é de 13 caracteres,temos

... c a b r a c a d a b r a r r a r r a d ...

c a b r a c a | d a b r a r <0,0,C(d)>

o=7

a b r a c a d | a b r a r r <7,4,C(r)>

l=4

o=3

a d a b r a r | r a r r a d <3,5,C(d)>

l=5

<0,0,C(d)> c a b r a c a | d

o=7

<7,4,C(r)> c a b r a c a d | a b r a r

l=4

o=3

<3,5,C(d)> c a b r a c a d a b r a r | r a r r a d

l=5

- Exemplo 2:

a b c a b c a b c a b c a b c d ...

<0,0,C(a)>,<0,0,C(b)>,<0,0,C(c)>,<3,12,C(d)>,...

- PKZIP, ZIP, LHarc, PNG, gzip e ARJ usam LZ77 + Códigos de comprimento variável.

3.2 – LZ78

- Cada nova frase é codificada como um índice de uma palavra do dicionário $+$ um caracter de inovação.
  - Dicionário incial: vazio
  - Codifico: palavra do dicionário $+$ inovação
  - Atualizo dicionário: última palavra codificada

a a a b b a b a a b a a a b a b

a | a a | b | b a | b a a | b a a a | b a b

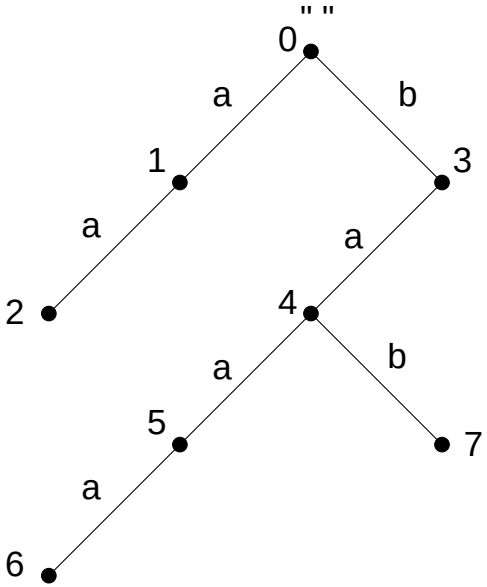
(0,a) (1,a) (0,b) (3,a) (4,a) (5,a) (4,b)

Dicionário:

0: " "
1: a
2: aa
3: b
4: ba
5: baa
6: $baaa$
7: bab

a | a a | b | b a | b a a | b a a a | b a b

(0,a) (1,a) (0,b) (3,a) (4,a) (5,a) (4,b)



- Codificação:
  - Vou andando na árvore, até que não é mais possível o casamento.
  - Não havendo casamento, incluo o nó na árvore correspondente ao próximo caracter.
  - É rápida.
- Decodificação:
  - Os símbolos recebidos dão o caminho na árvore e a sua construção.
  - É rápida.

3.3 – LZW

- Cada nova frase é codificada como um índice de uma palavra do dicionário, Não há caracter de inovação.
- Para isto, o dicionário inicial contém o alfabeto.
  - Dicionário incial: alfabeto.
  - Codifico: palavra do dicionário.
  - Atualizo dicionário: última palavra codificada $+1^{o}$ caracter da próxima palavra.

a a b a b a b a a a

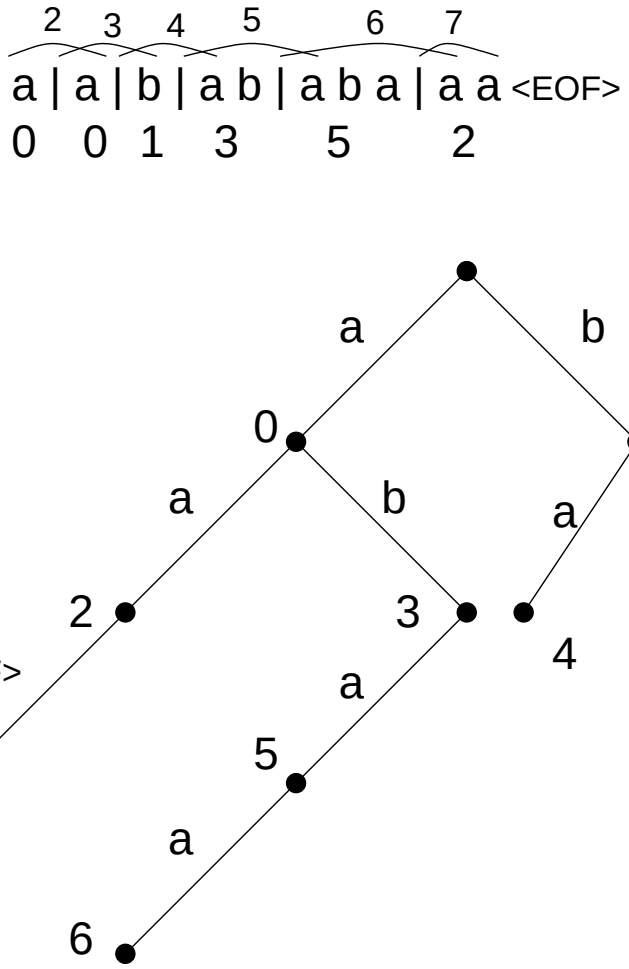
2 3 4 5 6 7

a | a | b | a b | a b a | a a <EOF>

0 0 1 3 5 2

Dicionário:

0: a
1: b
2: aa
3: ab
4: ba
5: aba
6: $abaa$
7: aa



- Uso da árvore: semelhante ao do LZ78.