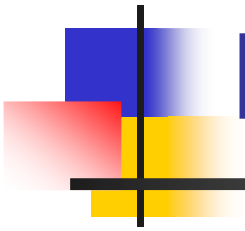
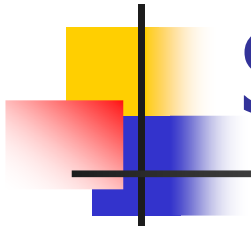


# SCC661 – Multimídia e Hipermídia

**Aula 2**

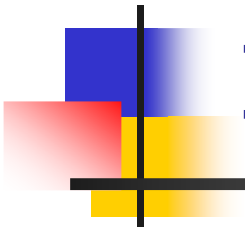
Prof. Dr. Marcelo Manzato
(mmanzato@icmc.usp.br)

Instituto de Ciências Matemáticas e de Computação - ICMC
Sala 3-160

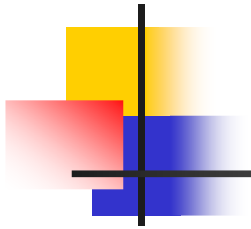


# Sumário

- 1. Princípios de Compressão.
- 2. Compressão de Texto.



# 1. Princípios de Compressão



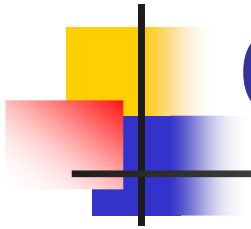
# 1.1 Por quê Comprimir?

- Preencher o "gap" demanda x capacidade
  - Usuários têm demandado aplicações com mídias cada vez mais sofisticadas.
  - Meios de transmissão e armazenamento são limitados.
    - Livro de 800 páginas. Cada página com 40 linhas. Cada linha com 80 caracteres. -> 800 * 40 * 80 = (1 byte por caracter) 2,44 MB.
    - Vídeo digital "full HD" (1920x1080p):
      - 1 segundo = 1.5 Gbits.
      - 2 horas = 1350 GBytes = 288 DVDs!
  - "Compressão vai se tornar redundante em breve, conforme as capacidades de armazenamento e transmissão aumentem."
    - Esta frase tem sido repetida nos últimos 20 anos.

# 1.2 Codificador Fonte e Decodificador Destino

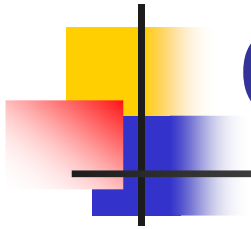
- Em sistemas multimídia, freqüentemente, a informação é comprimida antes de ser armazenada ou transmitida.
  - Algoritmo de compressão: principal tarefa do codificador fonte.
  - Algoritmo de descompressão: principal tarefa do decodificador destino.
- Implementação do algoritmo de compressão:
  - Em software: quando o tempo para compressão/descompressão não é crítico.
  - Em hardware: quando o tempo para compressão/descompressão é crítico.

# 1.3 Tipos de Algoritmos de Compressão

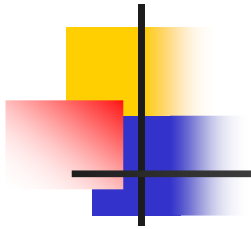


- Compressão sem perdas: Lossless.
  - Não há perda de informação no processo compressão/descompressão.
    - A informação "descomprimida" é uma cópia exata da informação original.
  - É dita ser reversível.
  - Exemplo de mídia que exige compressão lossless: texto.
    - Outro exemplo?

# 1.3 Tipos de Algoritmos de Compressão



- Compressão com perdas: Lossy.
  - Existe perda de informação no processo compressão/descompressão.
    - A informação "descomprimida" não é uma cópia exata da informação original.
    - Tal informação, contudo, é **percebida** como uma cópia.
  - Alcançam maiores taxas de compressão.
    - Para isso, descartam alguma informação – a perda é aceitável.
    - Quanto maior a compressão, maior a perda. -> compromisso.
  - Exemplo?



# 1.4 Codificação por Entropia

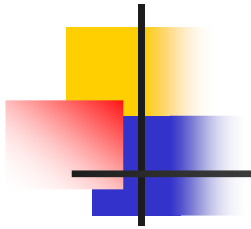
- É uma técnica de compressão sem perda (Lossless).
- É independente da informação sendo comprimida.
- Utilizada na codificação de várias mídias (representação).
- Origem:
  - Codificação estatística :: teoria da informação

# 1.4 Codificação por Entropia

- Entropia:
  - número médio mínimo de bits que são teoricamente necessários para "transmitir" um código da fonte de informações. (entropia da fonte)
  - Fórmula de Shannon:

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i$$

    - n = número de diferentes símbolos; $P_i$ = probabilidade de ocorrência do símbolo i.

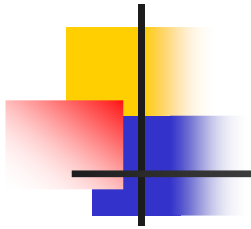


# 1.4 Codificação por Entropia

- Eficiência de um esquema de compressão:
  - Taxa da entropia da fonte comparado ao número médio de bits por código do esquema.
    - Quanto mais próximo o segundo estiver do primeiro, melhor o esquema.
  - Número médio de bits por código:

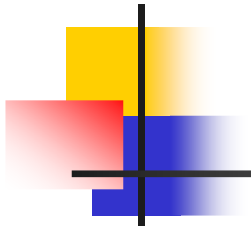
$$\sum_{i=1}^{n} N_i P_i$$

n = no. símbolos
$N_i$ = no. caracteres/código
$P_i$ = probabilidade



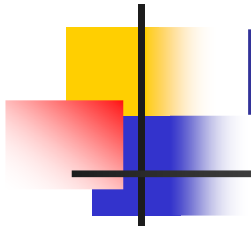
# 1.4 Codificação por Entropia

- Exemplo.
  - Novo método de compressão. Alfabeto: M, F, Y, N, 0 e 1. Freqüência: 0.25, 0.25, 0.125, 0.125, 0.125 e 0.125. Códigos: M = 10, F = 11, Y = 010, N = 011, 0 = 000, 1 = 001.
  - A) Qual a entropia da fonte?
  - B) Qual o número médio de bits por código?



# 1.4 Codificação por Entropia

- Qual a eficiência desse novo método de compressão?
  - Considerando que H=2.5 e M=2.5



# Exercício

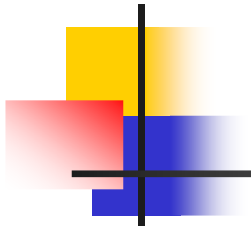
- Alfabeto:
  - A → 0.5 → 0
  - B → 0.25 → 10
  - C → 0.05 → 110
  - D → 0.2 → 111
- Calcule:
  - A entropia da fonte
  - Número médio de bits por código

# 1.4 Codificação por Entropia

- Codificação por entropia envolve:
  - Codificação estatística +
  - Codificação por carreira +
  - Codificação por diferenças

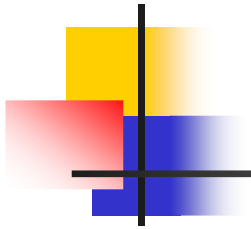
# 1.4 Codificação por Entropia

- Codificação estatística
  - Baseada na freqüência dos símbolos.
    - Símbolos com maior freqüência = menor código.
  - Propriedade do prefixo.
    - Um código não pode ser prefixo de um código mais longo.
    - Ex.: a=01, b=10, c=110, d=101
    - 011010110 → abdb
  - Ex.: Códigos de Huffman.



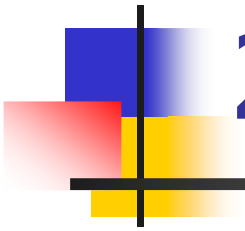
# 1.4 Codificação por Entropia

- Codificação *Run-Length*
  - *Run-Length* = carreira
  - Útil para codificar informação composta por uma longa substring do mesmo caractere ou dígito binário.
  - Indica o número de caracteres ou bits na substring.
  - 00000111111100001111111100...
    - 0,5 1,7 0,4 1,10 0,2 ... (0,5 é um código – *codeword*)
    - 5, 7, 4, 10, 2 ... (5, 7, etc. são códigos)
  - Número de bits por código (assumindo tamanho fixo):
    - É determinado pelo comprimento da maior substring.

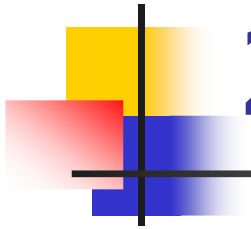


# 1.4 Codificação por Entropia

- Codificação por diferença
  - Usado quando a amplitude de um sinal pode assumir valores em uma faixa larga, mas a diferença entre valores consecutivos é pequena.
  - Codifica a diferença.
    - Primeiro valor.
  - Pode ocasionar ou não perdas.
    - Depende do número de bits usados para armazenar a maior diferença entre amplitudes sucessivas.
  - Exemplo:
    - 33200, 33100, 33050, 33152, ...
    - 33200, -100, -50, 102, ...



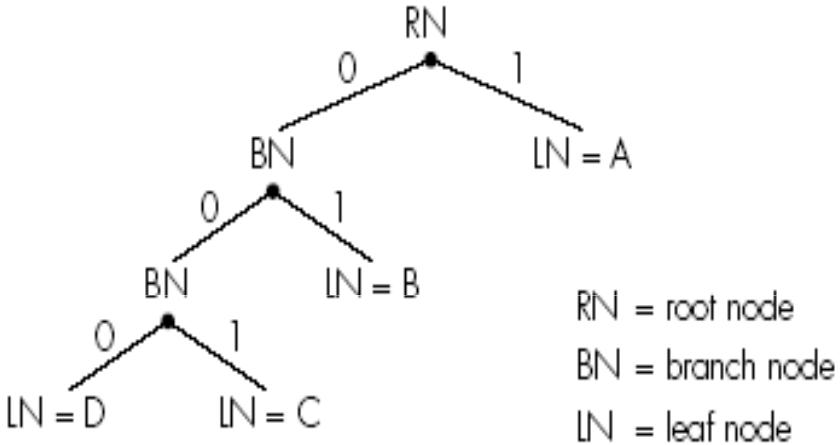
# 2. Compressão de Texto



# 2.1 Codificação de Huffman

- Texto a ser codificado é primeiro analisado para extrair a freqüência relativa dos caracteres.
- A codificação envolve a criação de uma árvore binária não-balanceada.
  - Os caracteres estão nas folhas.
- Tal árvore é chamada árvore de Huffman.

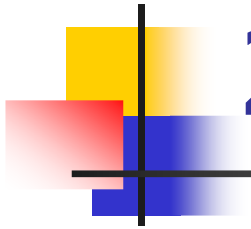
# 2.1 Codificação de Huffman




A = 1
B = 0 1
C = 0 0 1
D = 0 0 0

- A árvore acima corresponde à string AAAABBCD.
  - Códigos de Huffman requerem, nesse caso, 14 bits.
  - Versus 64 bits (8 caracteres * 8 bits) do código ASCII.

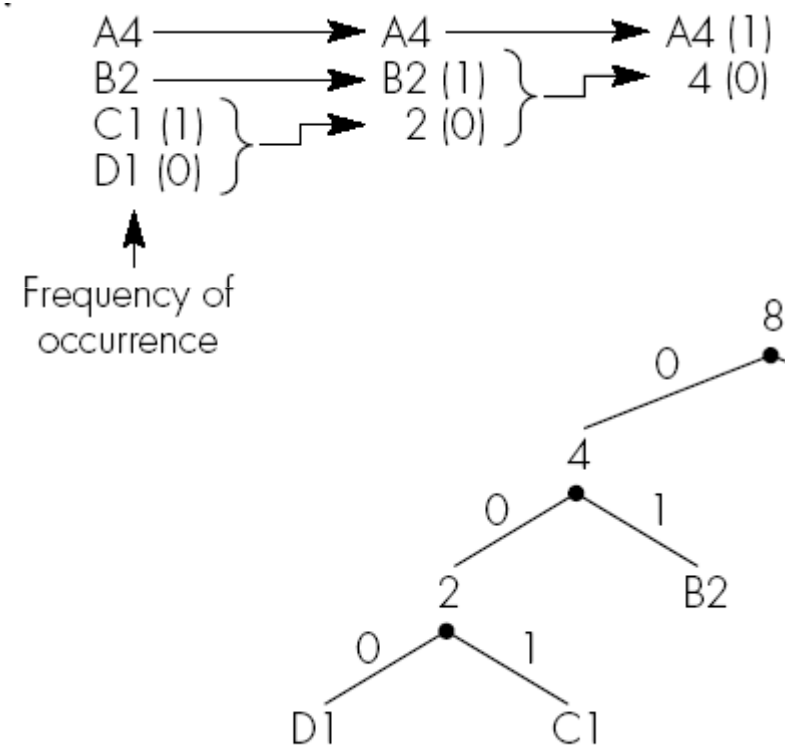
- Na árvore de Huffman, aresta à direita de um nó recebe valor 1; e a aresta à esquerda recebe valor 0.
- Percorrendo-se a árvore da raiz em direção às folhas obtém-se os códigos de cada caractere.



# 2.1 Codificação de Huffman

- Para construir uma árvore de Huffman é necessário obter a freqüência dos caracteres:
  - Os caracteres com maior freqüência devem ter os menores códigos.
  - Monta-se uma lista ordenada pela freqüência:
    - A4
    - B2
    - C1
    - D1

# 2.1 Codificação de Huffman

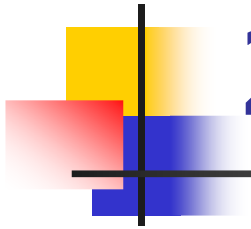



A4 = (1) → 1
B2 = (1) (0) → 0 1
C1 = (1) (0) (0) → 0 0 1
D1 = (0) (0) (0) → 0 0 0

Starting at leaf node

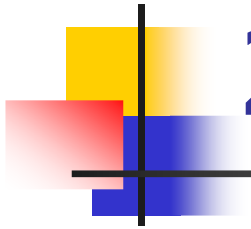
Starting at root node

Weight order = D1 C1 2 B2 4 A4 8 ✔



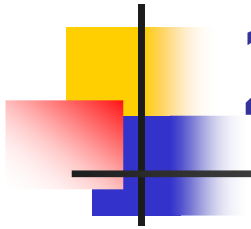
# 2.1 Codificação de Huffman

- Árvore ótima (de Huffman):
  - Basta verificar, da esquerda para a direita, e de baixo para cima, se os pesos estão em ordem crescente.
- Árvore de Huffman tem a propriedade do prefixo.
  - Nenhum código é prefixo de outro código.



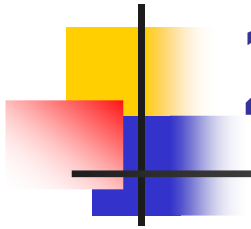
# 2.1 Codificação de Huffman

- Codificação:
  - Basta substituir os caracteres pelos respectivos códigos.
  - No exemplo, a string AAAABBCD será codificada como:
    - 11110101001000



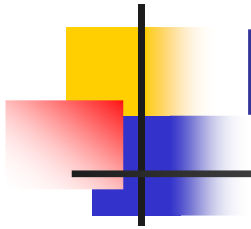
# 2.1 Codificação de Huffman

- Decodificação:
  - Basta usar o código de Huffman como índice para percorrer a árvore.
  - No exemplo, o primeiro bit do código 111101010010000 é 1. A partir da raiz da árvore, percorre-se a mesma à direita (1). Se encontrou um nó folha, escreve o caracter correspondente, volta para raiz e pega próximo bit do código. Senão, pega próximo bit do código.



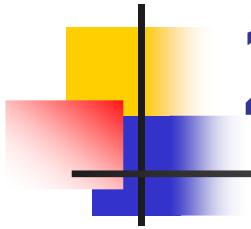
# 2.1 Codificação de Huffman

- Observações:
  - Ambos, codificador e decodificador devem conhecer a tabela (ou árvore) de códigos.
  - Se a tabela é enviada/codificada junto com os dados, ocorre overhead.
  - O decodificador pode conhecer a tabela com antecedência.
    - Análise estatística do uso dos caracteres em uma determinada língua.
    - Esse método não é exato.
      - Alguns textos não vão atingir o máximo de compressão que poderiam.



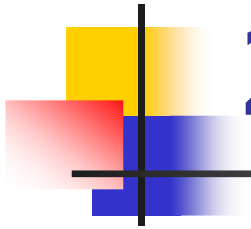
# Exercício

- Seja uma tabela de freqüências relativas como segue:
  - A e B = 0,25; C e D = 0,14; E, F, G e H = 0,055.
- Derive um conjunto de códigos usando o método de Huffman. Construa a árvore e verifique que ela é ótima.
- Derive o número médio de bits por caractere de seu código e compare com:
  - A entropia da fonte.
  - Um código binário de tamanho fixo.
  - Códigos ASCII de 7 bits.



# 2.2 Codificação Aritmética

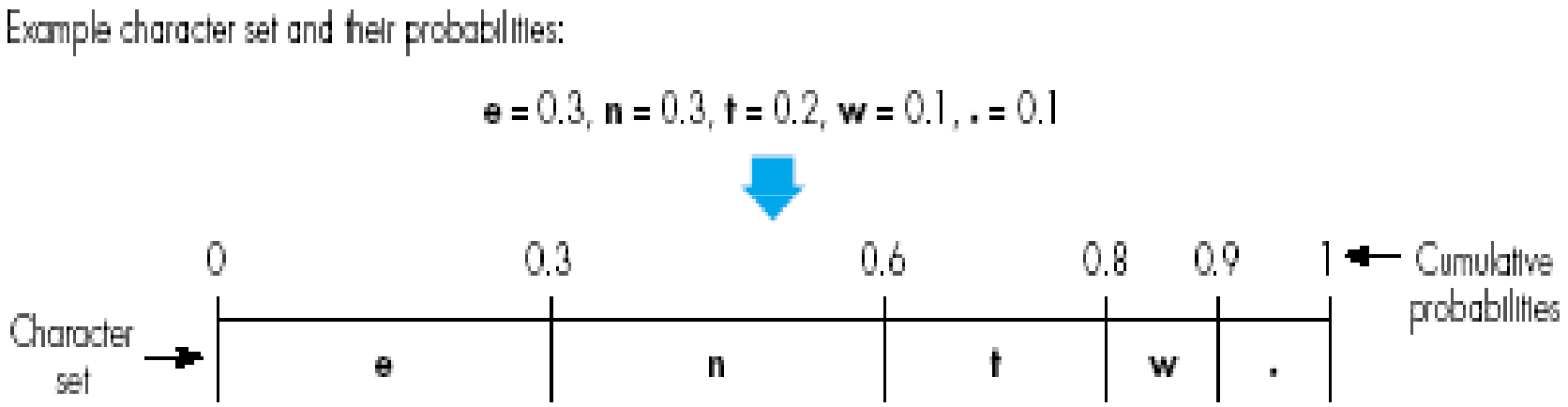
- Método de Huffman atinge o valor da Entropia apenas em algumas situações.
  - Depende da probabilidade de aparecimento dos caracteres no texto.
- Codificação Aritmética sempre atinge o valor da Entropia.
  - Mais complexa que Huffman.
  - Iremos estudar apenas o modo básico.



# 2.2 Codificação Aritmética

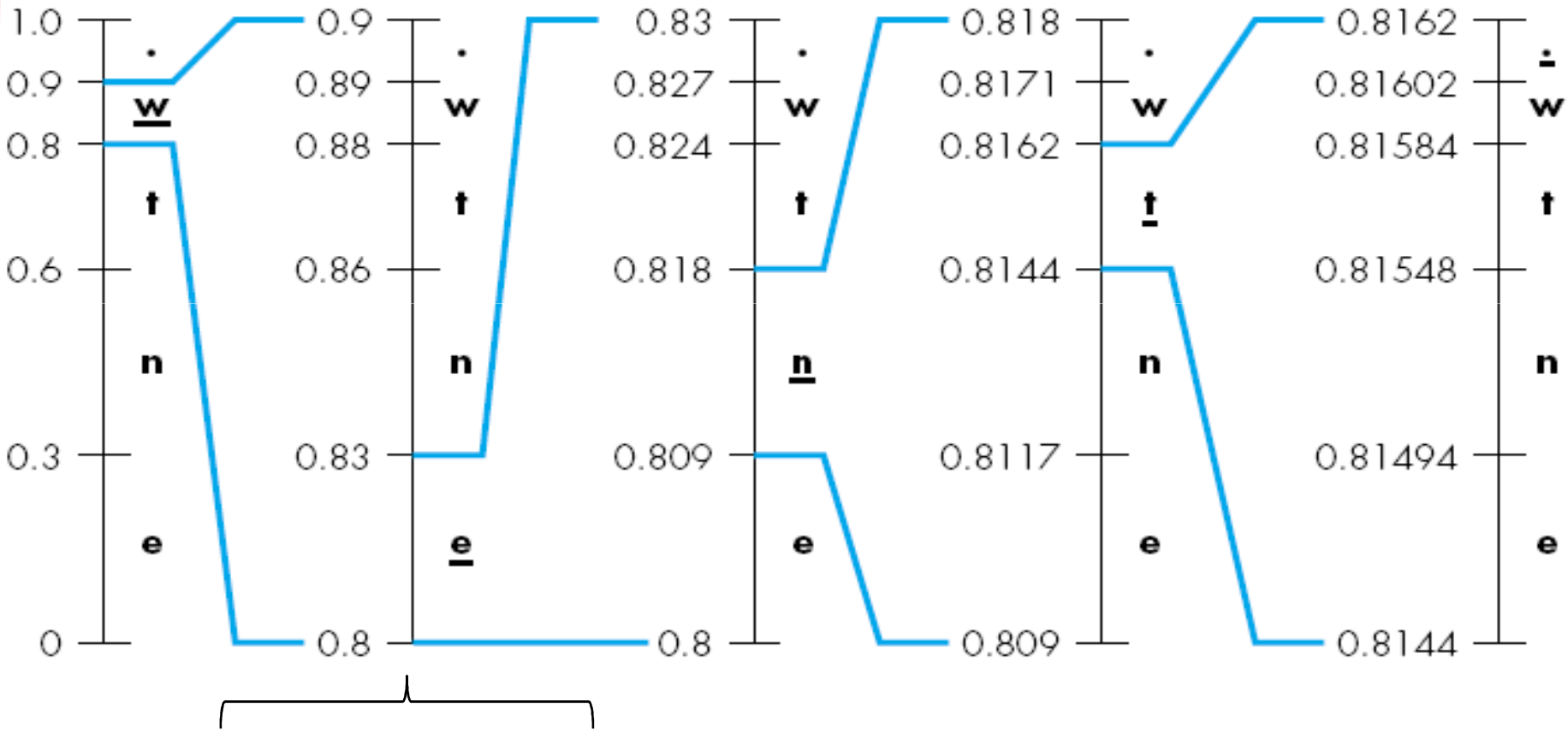
- String a ser codificada: `went.`
- Probabilidades:
  - e = 0,3; n = 0,3; t = 0,2; w = 0,1; . = 0,1
  - . = terminador de string
- Conjunto de caracteres deve ser dividido no intervalo de 0 a 1, respeitando-se a proporção das probabilidades.

# 2.2 Codificação Aritmética



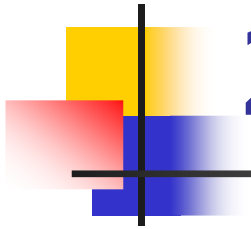

- Cada subintervalo, na ordem da mensagem, é subdividido respeitando-se as proporções.

# 2.2 Codificação Aritmética




e=0.8+(0.3*(0.9-0.8))=0.83
n=0.83+(0.3*(0.9-0.8))=0.86
t=0.86+(0.2*(0.9-0.8))=0.88
w=0.88+(0.1*(0.9-0.8))=0.89
.=0.89+(0.1*(0.9-0.8))=0.9

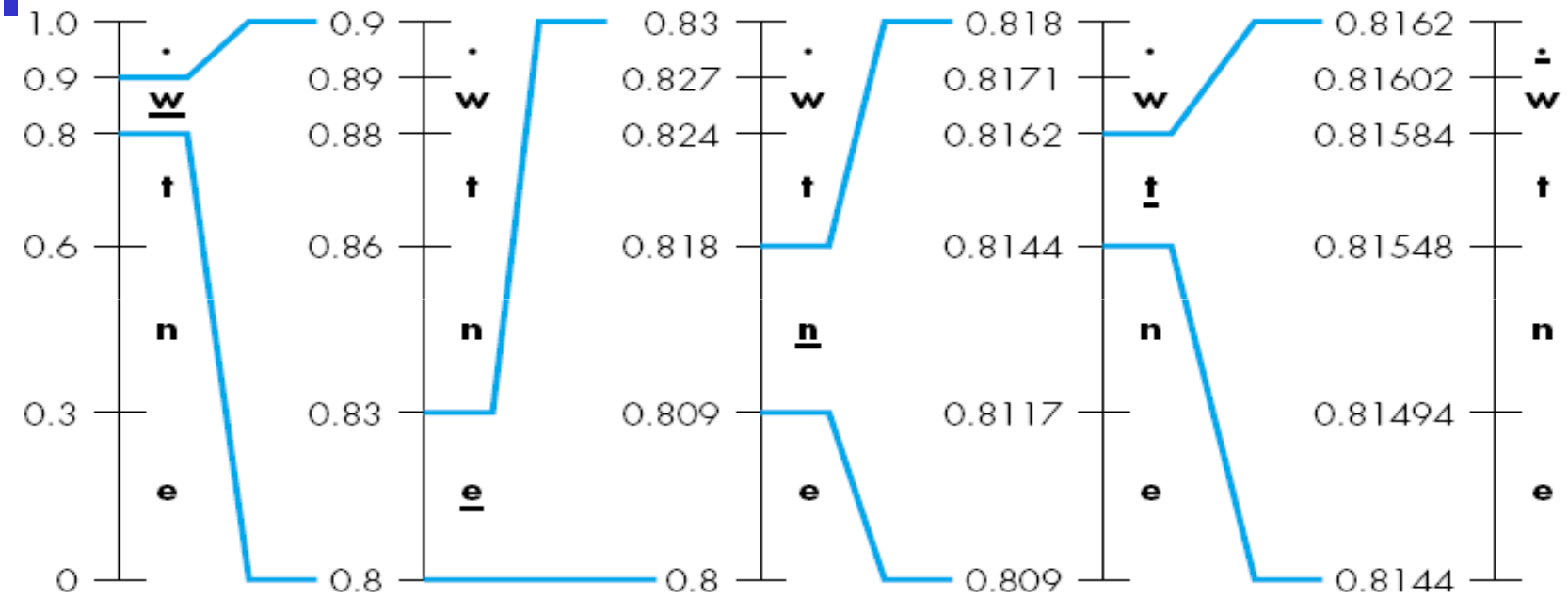
e = 0,3; n = 0,3; t = 0,2; w = 0,1; . = 0,1



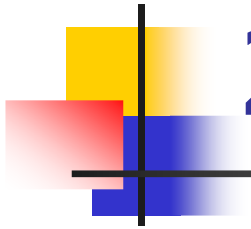
# 2.2 Codificação Aritmética

- Nesse exemplo, o código pode ser qualquer número entre 0,81602 e 0,8162.
  - 0,8161, por exemplo.
- Decodificador conhece o alfabeto, as probabilidades e os intervalos.
  - Então pode seguir o mesmo processo do codificador para decodificar a mensagem 0,8161.

# 2.2 Codificação Aritmética

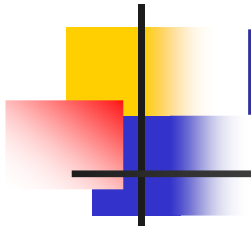



0,8161 => primeiro caracter é "w", pois 0,8161 está no intervalo 0,8-0,9. O segundo é "e", pois 0,8161 está no intervalo 0,8-0,83. E assim por diante.



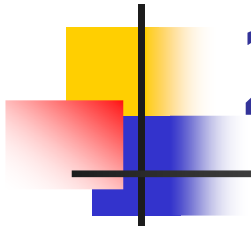
# 2.2 Codificação Aritmética

- Nesse método, o número de dígitos no código cresce linearmente de acordo com o tamanho da string.
- Logo, o número máximo de caracteres em uma string é determinado pela precisão de ponto flutuante na máquina destino.
  - Strings grandes podem ser quebradas em duas ou mais substrings.



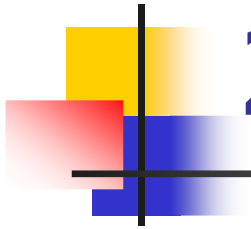
# Exercício

- Considerando que:
  - A=0.4, B=0.3, C=0.2 e D=0.1
- Codifique a seguinte string usando a codificação aritmética:
  - “CADBC”



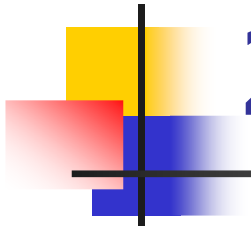
# 2.3 Lempel-Ziv (LZ)

- Técnica baseada em dicionário.
  - Em vez de codificar caracteres, codifica strings, as quais são armazenadas em uma tabela (dicionário).
  - Codificação apenas do índice da tabela
- Exemplo:
  - Dicionário de 25000 palavras → $2^{15}$ → 15 bits para codificação
  - 7-bit ASCII: "multimídia" → 10 * 7 = 70 bits
  - Razão de compressão:  4.7 : 1



# 2.3 Lempel-Ziv-Welsh (LZW)

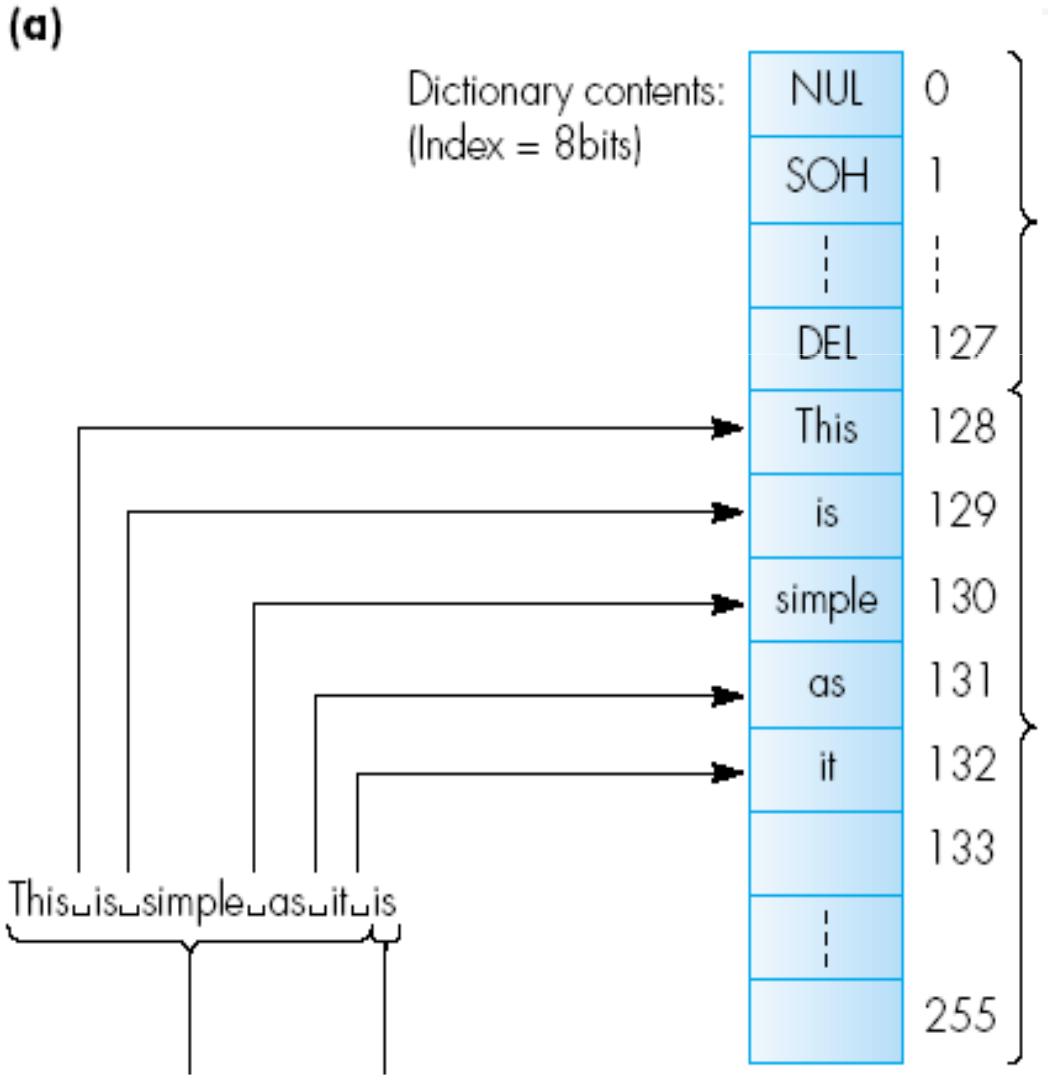
- Baseada no algoritmo Lempel-Ziv.
- Técnica baseada em dicionário.
  - Em vez de codificar caracteres, codifica strings, as quais são armazenadas em uma tabela (dicionário).
- O Codificador e o decodificador constroem o dicionário dinamicamente.

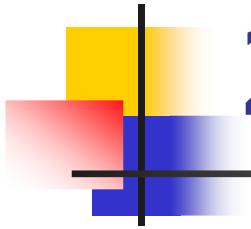


# 2.3 Lempel-Ziv-Welsh (LZW)

- Inicialmente os dicionários contém apenas a tabela ASCII.
  - Cada caractere já contém um índice
- A primeira palavra é codificada usando os índices dos caracteres que a compõe.
  - A palavra é armazenada no dicionário.
  - Na próxima ocorrência o codificador envia seu código.
  - No exemplo, usa-se o espaço em branco como delimitador entre palavras.

# 2.3 Lempel-Ziv-Welsh (LZW)



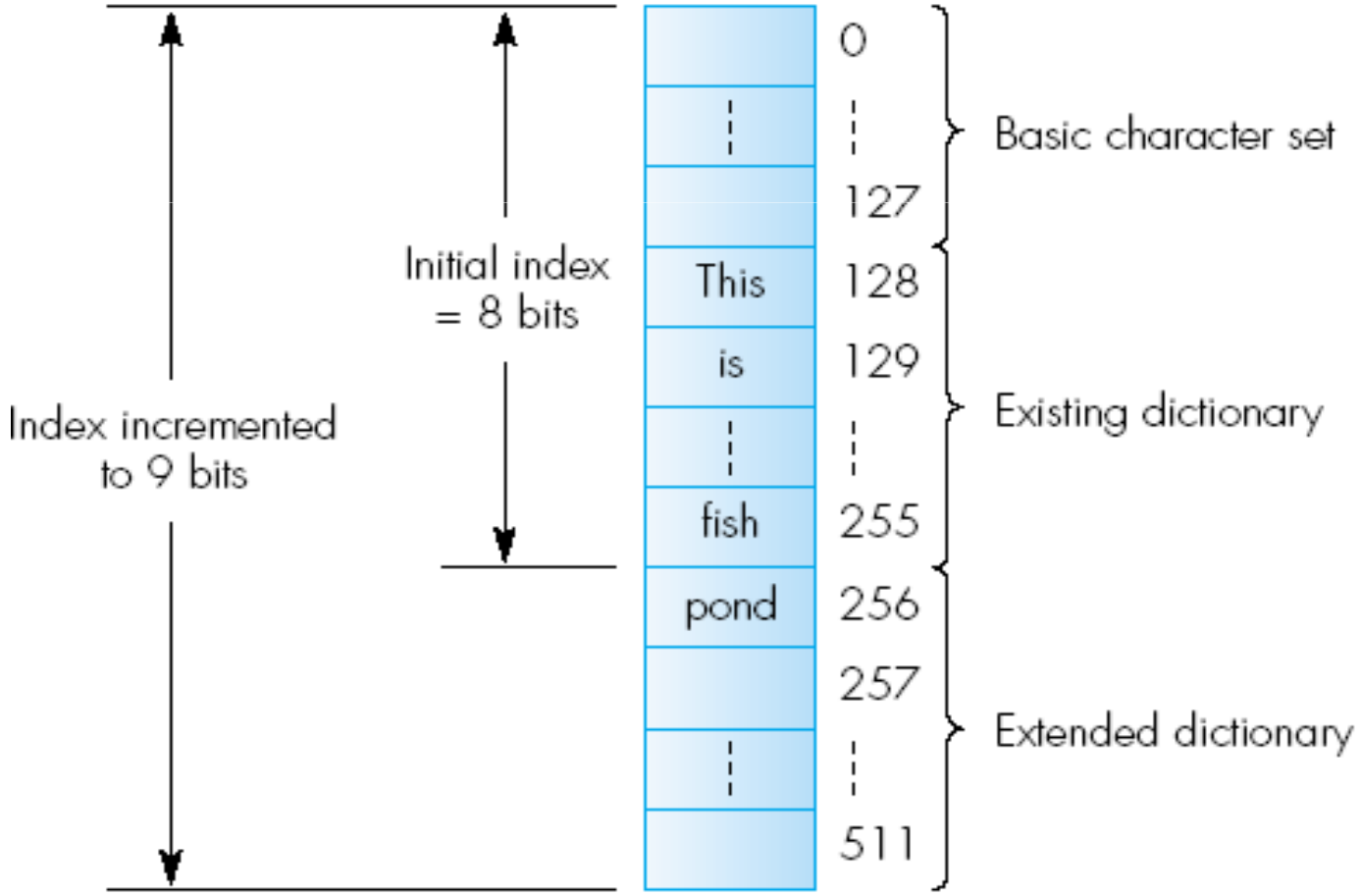


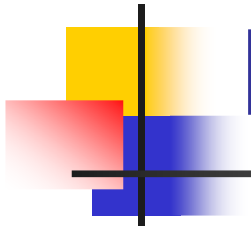
# 2.3 Lempel-Ziv-Welsh (LZW)

- Questão chave para compressão:
  - Número de entradas no dicionário.
    - Determina o número de bits para os índices.
- Poucas entradas:
  - Textos pequenos.
- Muitas entradas:
  - Entradas vazias.
  - Códigos com muitos bits.
    - Menos compressão.

# 2.3 Lempel-Ziv-Welsh (LZW)

- Dicionário dinâmico.





# Para Saber Mais

- Halsall, F. Multimedia Communications: Applications, Networks, Protocols, and Standards, Addison-Wesley Publishing, 2001. ISBN: 0201398184. Capítulo 3.
- Mandal, M. K. Multimedia Signals and Systems. Kluwer Academic Publishers, 2002. ISBN: 1402072708. Capítulo 6.