Técnicas de Compactação e Compressão

Profa. Débora Christina Muchaluat Saade

debora@midiacom.uff.br

Técnicas de Compactação e Compressão

- ✓ **Compactação X Compressão**
- ✓ **Classificação das técnicas de compressão**
  - ***Codificação por Entropia, na Origem e Híbrida***
- ✓ **Técnicas de Compactação**
  - ***Codificação por carreira***
  - ***Codificação por Shannon-Fano***
  - ***Codificação de Huffman***
  - ***Codificação de Lempel-Ziv-Welch (LZW)***
  - ***Codificação aritmética***

Técnicas de Compactação e Compressão

- ✓ **Técnicas de Compressão**
  - ***Redução do domínio***
  - ***Redução do espaço de quantização***
  - ***Codificação preditiva***
  - ***Codificação por sub-bandas***
  - ***Codificação por transformadas***
  - ***Quantização vetorial***

Compactação x Compressão

- **Compactação:**
  - ***Quando eliminamos apenas a redundância de um sinal***
  - ***Não há perda de informação***
  - ***Compressão sem perdas***
  - ***Podem ser usadas para qualquer sinal (mídia)***
- **Compressão:**
  - ***Quando, na redução dos dados, há perda de informação***
  - ***Compressão com perdas***
  - ***Algumas técnicas são usadas em sinais específicos***
  - ***Compressão perceptualmente sem perdas***
    - humanos não percebem
    - Ex.: MP3 para áudio

Classificação das Técnicas de Compressão

- ✓ **Codificação por Entropia**
  - ***trata cadeias de bits sem levar em conta seu significado***
  - ***técnica genérica, sem perda e totalmente reversível***
    - Ex.: técnicas de compactação: codificação por carreira, codificação de Huffman, codificação aritmética
- ✓ **Codificação na Origem**
  - ***leva em consideração a semântica dos dados***
  - ***processa o dado original distinguindo o dado relevante e o irrelevante***
    - removendo dados irrelevantes comprime o dado original
    - Ex.: técnicas de compressão: codificação preditiva, codificação por sub-bandas, codificação por transformadas, quantização vetorial
- ✓ **Codificações Híbridas**
  - ***Combinam técnicas com e sem perdas (várias técnicas são agrupadas para formar uma nova técnica de codificação)***
    - Ex.: JPEG, MPEG, H.263, ...

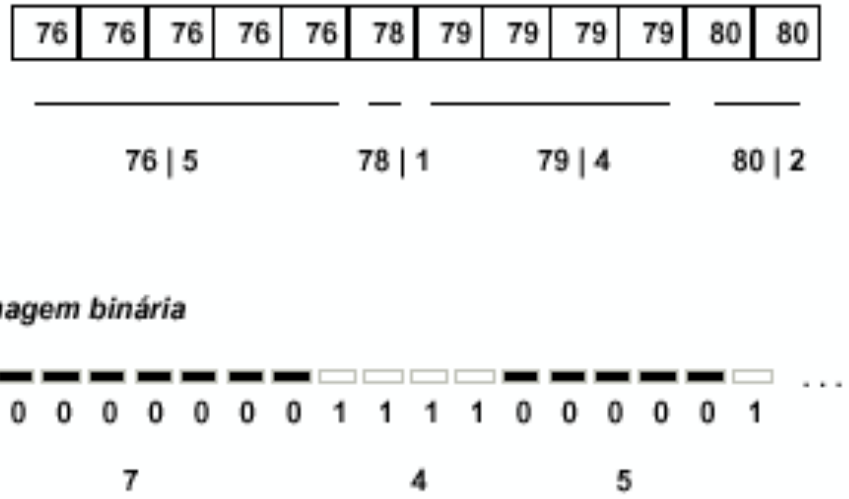
Técnicas de Compactação

Profa. Débora Christina Muchaluat Saade

debora@midiacom.uff.br

Compactação

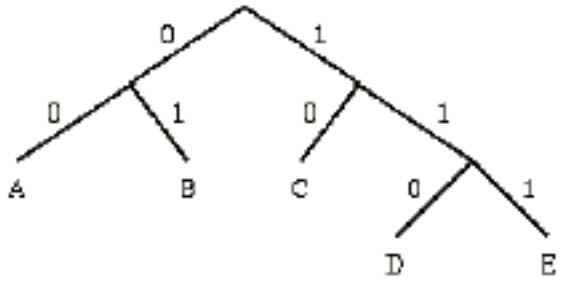
- ✓ **Codificação por carreira (Run Length Coding)**



- ✓ **Técnica boa quando informação se repete**
- ✓ **OBS.: Toda técnica de compactação pode diminuir ou aumentar o volume de dados**

Codificação por Shannon-Fano

- ✓ **A seqüência a ser compactada deve ser analisada previamente, identificando-se os caracteres e suas respectivas freqüências/ probabilidades**
- ✓ **Ordene os símbolos de acordo com suas freqüências. Ex.: ABCDE**
- ✓ **Divida recursivamente em 2 partes, cada uma com aproximadamente o mesmo número de contagem**



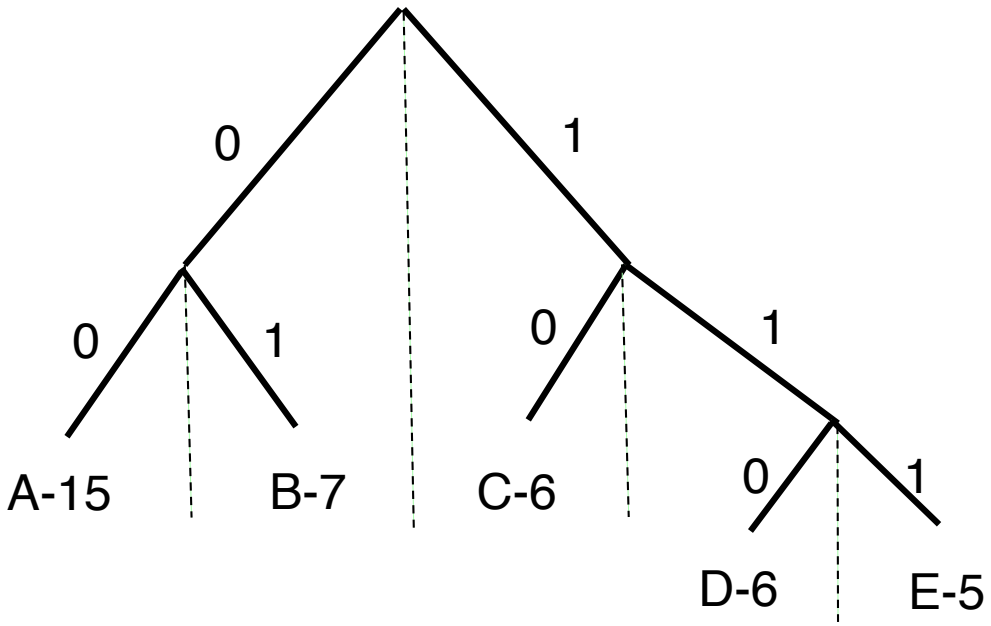
Símbolo	Freqüência	Código	Subtotal (#bits)
A	15	00	30
B	7	01	14
C	6	10	12
D	6	110	18
E	5	111	15

Total(# de bits) = 89

Compactação = 117 : 89

Exemplo – Codificação por Shannon-Fano

- ✓ **Frequência dos símbolos:**
  - *Símbolo – Frequência – Código – Número de bits*
  - *A – 15 – 00 – 30*
  - *B – 7 – 01 – 14*
  - *C – 6 – 10 – 12*
  - *D – 6 – 110 – 18*
  - *E – 5 – 111 – 15*



- ✓ **Taxa de Compactação**
  - *Num. bits original / num. bits compactado*
  - *(39 símbolos x 3 bits) / (30+14+12+18+15) = 117:89*

Codificação por Shannon-Fano

- ✓ **Sequência ABDEACD...**
  - *00 01 110 111 00 10 110 ...*
- ✓ **Envia:**
  - *Símbolos e palavras-código (codewords)*
    - A – 00
    - B – 01
    - C – 10
    - D – 110
    - E – 111
  - *Sequência compactada*

Codificação de Huffman

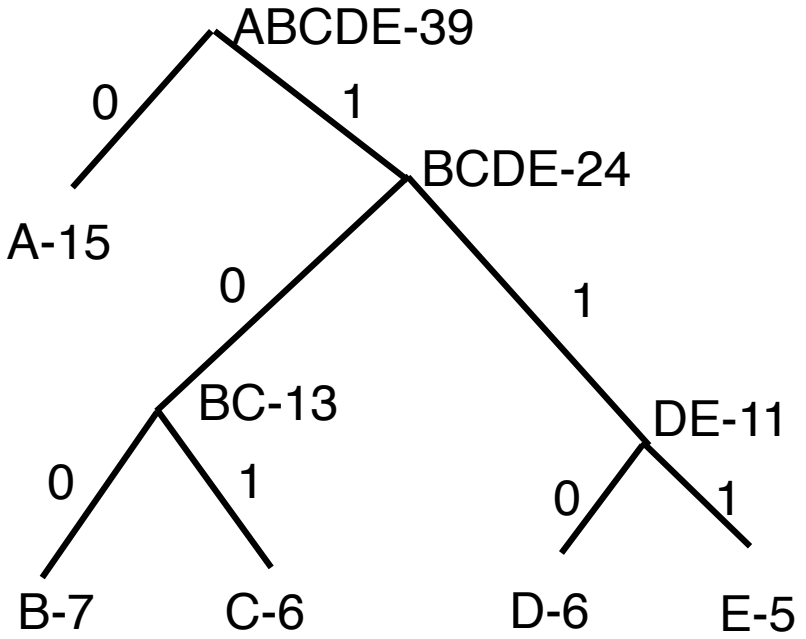
- ✓ **Codificação Estatística**
- ✓ **A sequência a ser compactada deve ser analisada previamente, identificando-se os caracteres e suas respectivas frequências/probabilidades**
- ✓ **Atribui menos bits a símbolos que aparecem mais frequentemente e mais bits para símbolos que aparecem menos**

Codificação de Huffman

- **Algoritmo:**
  - ***Insira os nós (símbolos/frequências) em uma lista***
  - ***Repita até que a lista contenha apenas um nó:***
    - Selecione os dois nós de mais baixa frequência e crie um nó pai para ambos
    - Atribua ao nó pai a soma das frequências e insira-o na lista
    - Atribua os códigos 0 e 1 aos dois ramos da árvore e retire os filhos da lista

Exemplo – Codificação de Huffman

- ✓ **Frequência dos símbolos:**
  - ***Símbolo – Frequência – Código – Número de bits***
  - ***Passo 1 - A15; B7; C6; D6; E5***
  - ***Passo 2 - A15; DE11; B7; C6***
  - ***Passo 3 - A15; BC13; DE11***
  - ***Passo 4 - BCDE24; A15***
  - ***Passo 5 - ABCDE39***

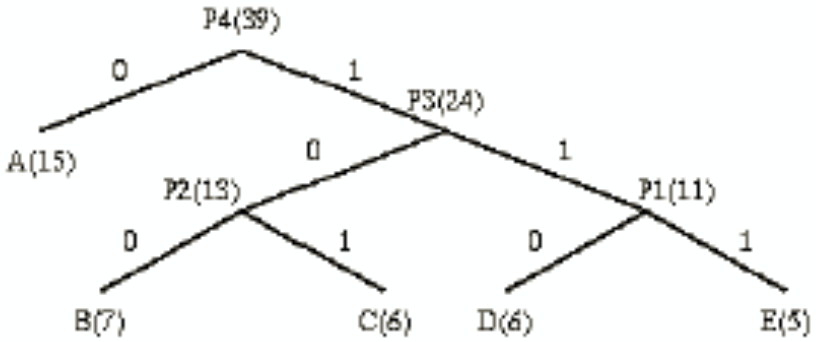


Exemplo - Huffman

Símbolo	Freqüência	Código	Subtotal (# de bits)
A	15	0	15
B	7	100	21
C	6	101	18
D	6	110	18
E	5	111	15

- ✓ **Sequência ABDEACD...**
  - *0 100 110 111 0 101 110 ...*
- ✓ **Taxa de Compactação**
  - *(39 símbolos x 3 bits) / (15+21+18+18+15) = 117:87*

Total (# de bits) = 87
Compactação = 117 : 87

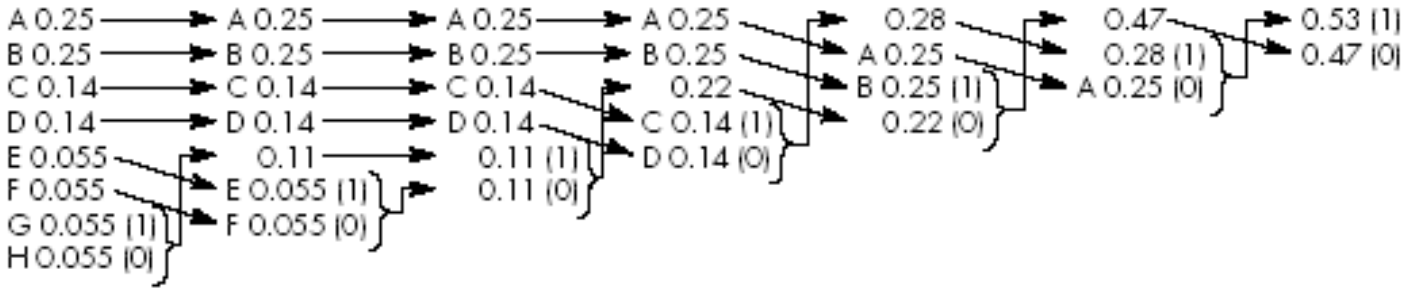


Exemplo - Huffman

- **Encontre os códigos de Huffman dos seguintes símbolos com as respectivas probabilidades de ocorrência:**
  - *A – 0,25*
  - *B – 0,25*
  - *C – 0,14*
  - *D – 0,14*
  - *E – 0,055*
  - *F – 0,055*
  - *G – 0,055*
  - *H – 0,055*

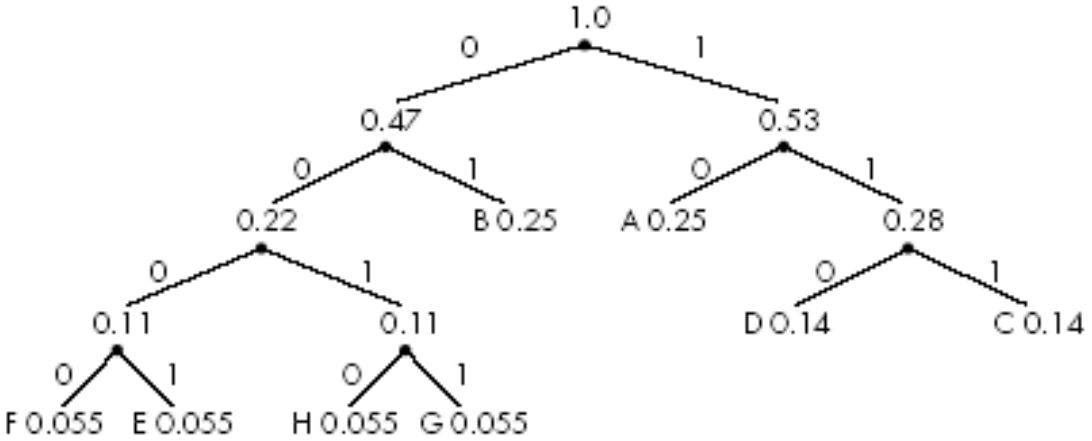
Exemplo - Huffman

(a)



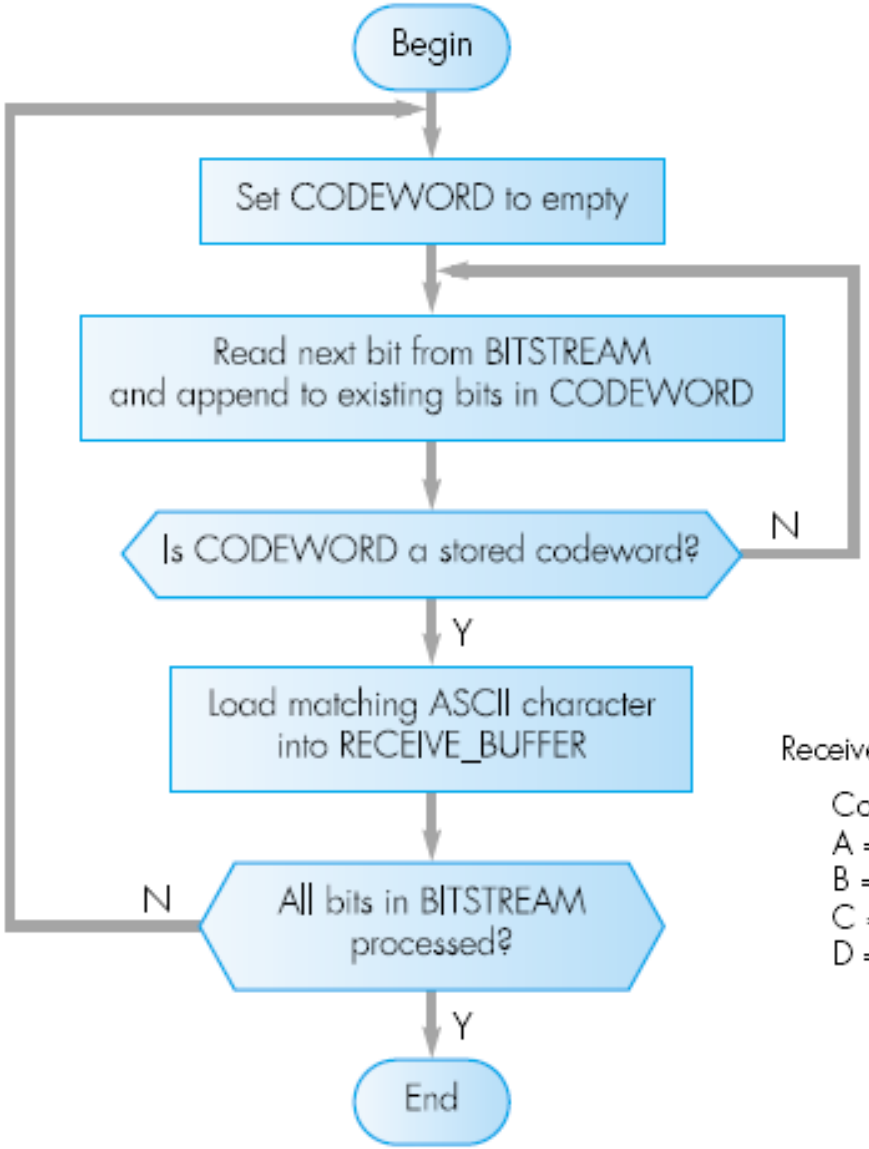
A = (0) (1) → 10
B = (1) (0) → 01
C = (1) (1) (1) → 111
D = (0) (1) (1) → 110
E = (1) (0) (0) (0) → 0001
F = (0) (0) (0) (0) → 0000
G = (1) (1) (0) (0) → 0011
H = (0) (1) (0) (0) → 0010

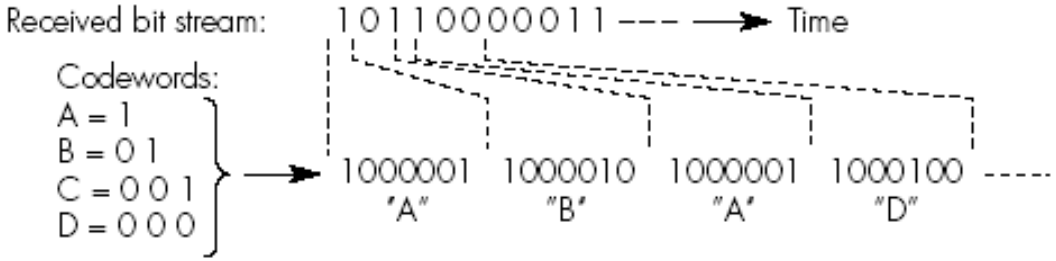
(b)



Weight order = 0.055 0.055 0.055 0.055 0.11 0.11 0.14 0.14 0.22 0.25 0.25 0.28 0.47 0.53 ✔

Huffman - Decodificação





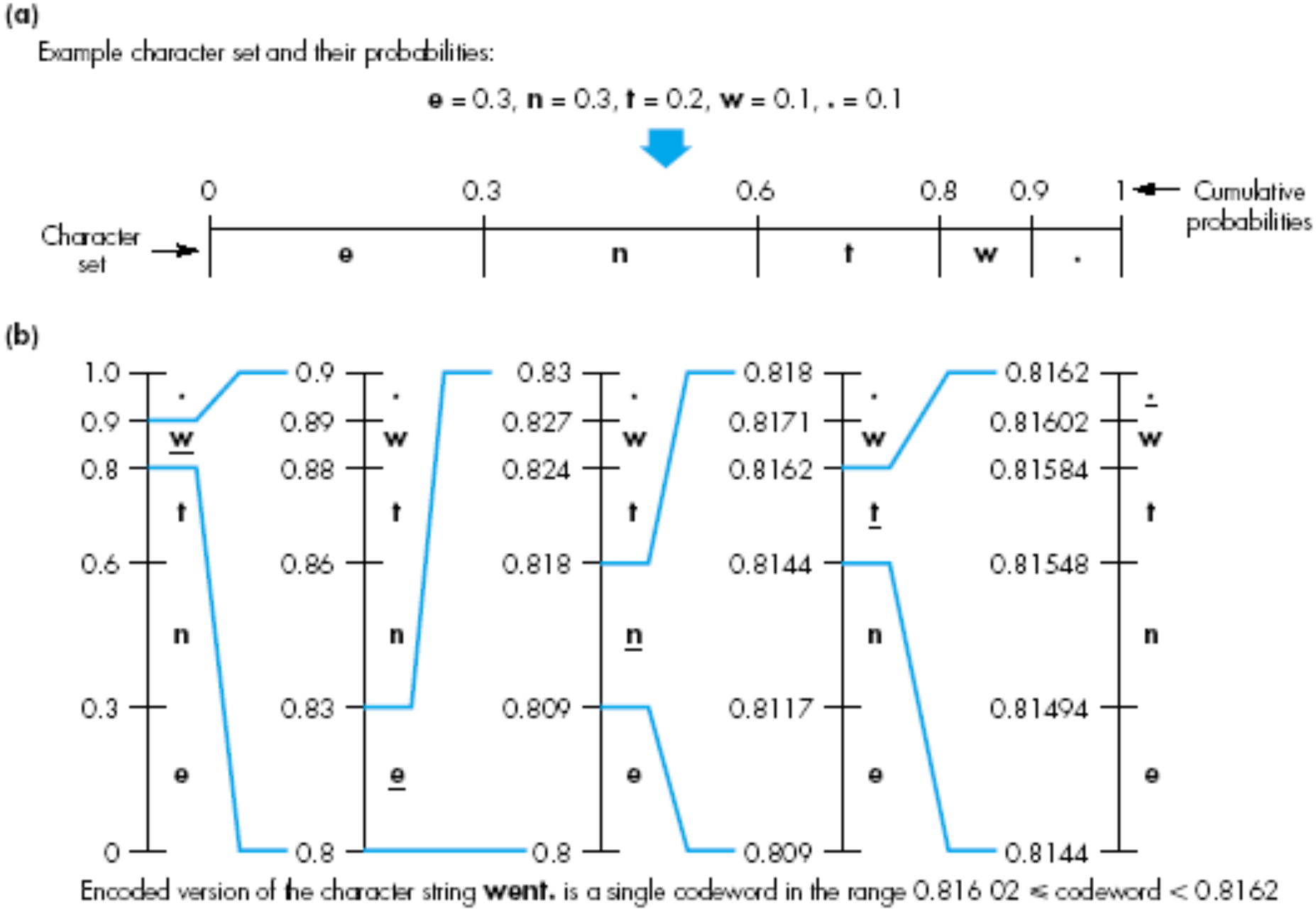
Codificação de Huffman

- ✓ **Nem todos os caracteres precisam ter uma representação codificada na tabela de Huffman**
  - ***apenas os caracteres com alta probabilidade de ocorrência***
  - ***demais são codificados diretamente e marcados com flag especial***
- ✓ **Técnica útil quando o número de caracteres diferentes é muito grande mas apenas alguns deles têm uma alta probabilidade de ocorrência**

Codificação Aritmética

- ✓ **Utiliza um único código por string de caracteres codificada**
- ✓ **A sequência a ser compactada deve ser analisada previamente, identificando-se os caracteres e suas respectivas frequências/ probabilidades**
  - *Ex.: e=0.3; n=0.3; t=0.2; w=0.1; .=0.1*
- ✓ **Precisa de um caracter marcando o fim de cada sequência (.)**
- ✓ **Algoritmo:**
  - ***Dividir o intervalo de [0, 1] de acordo com a probabilidade de ocorrência de cada caracter na sequência***
  - ***Repita até o caracter de fim de sequência***
    - Escolha o intervalo correspondente ao próximo caracter e divida-o novamente de acordo com as probabilidades iniciais
  - ***O código pode ser qualquer número dentro do último intervalo encontrado (ValorInicial <= código < ValorFinal)***

Exemplo – Codificação Aritmética



Encoded version of the character string **went.** is a single codeword in the range 0.816 02 ≤ codeword < 0.8162

Codificação Aritmética

- **O número de dígitos decimais no código aumenta linearmente com o número de caracteres na string**
  - ***O número máximo de caracteres em uma string é determinado pela precisão com a qual números de ponto flutuante são representados nos computadores de origem e destino***
  - ***Por essa razão, mensagens completas devem ser fragmentadas em várias strings menores e cada string deve ser codificada separadamente***

Codificação de Lempel-Ziv-Welch

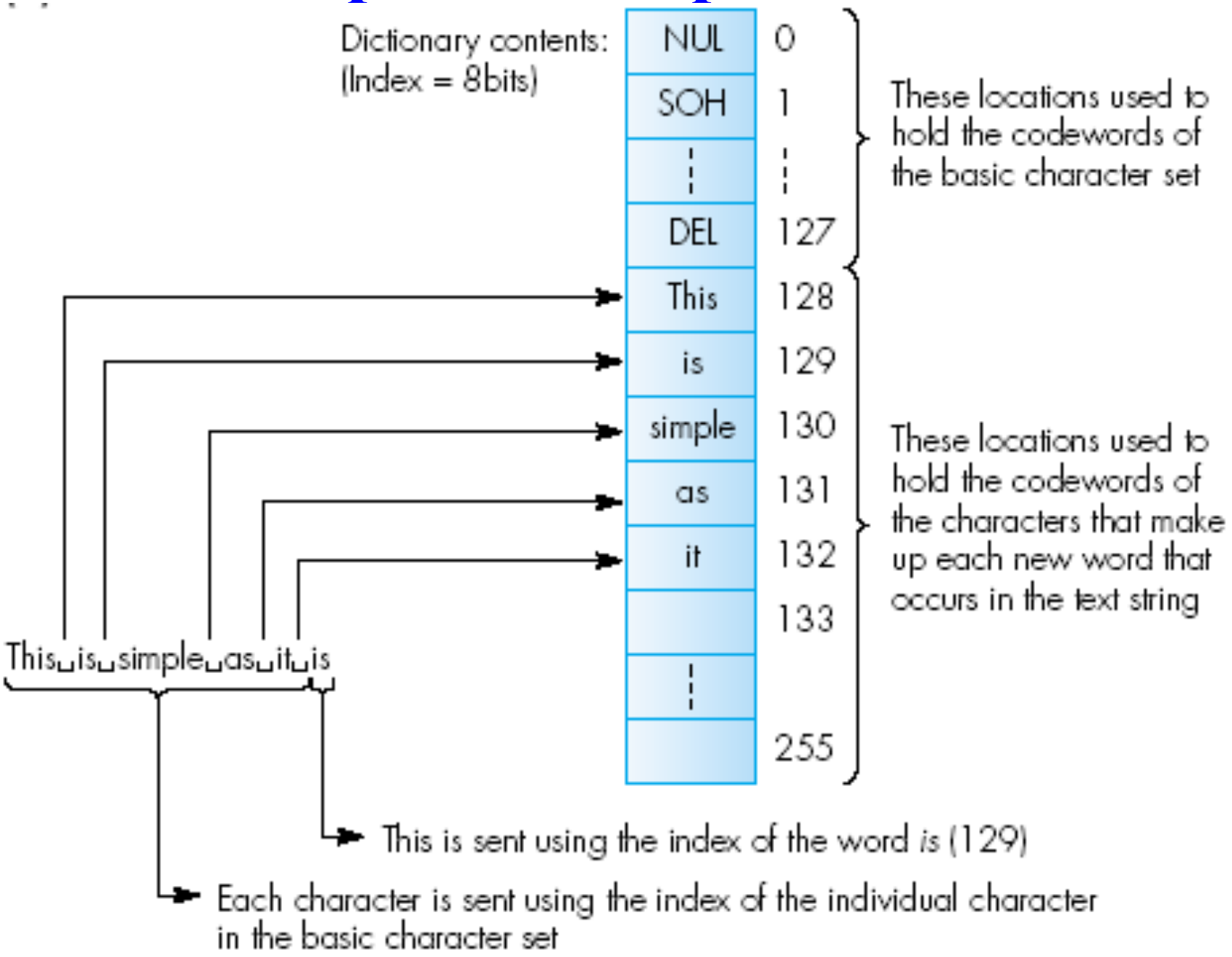
✓ Codificação de Lempel-Ziv-Welch (LZW)

- ***Ao invés de utilizar caracteres como a base da codificação, utiliza strings de caracteres***
- ***É baseada na construção de um dicionário de palavras (grupos de um ou mais caracteres) a partir do fluxo de entrada***
- ***Quando uma nova palavra é encontrada na sequência de entrada***
  - Ela é adicionada ao dicionário
- ***Se a palavra encontrada na sequência de entrada já foi registrada***
  - ela é substituída pelo código no dicionário
- ***Esta técnica é boa para compressão de arquivos textos, onde temos uma grande repetição de palavras de uso frequente***

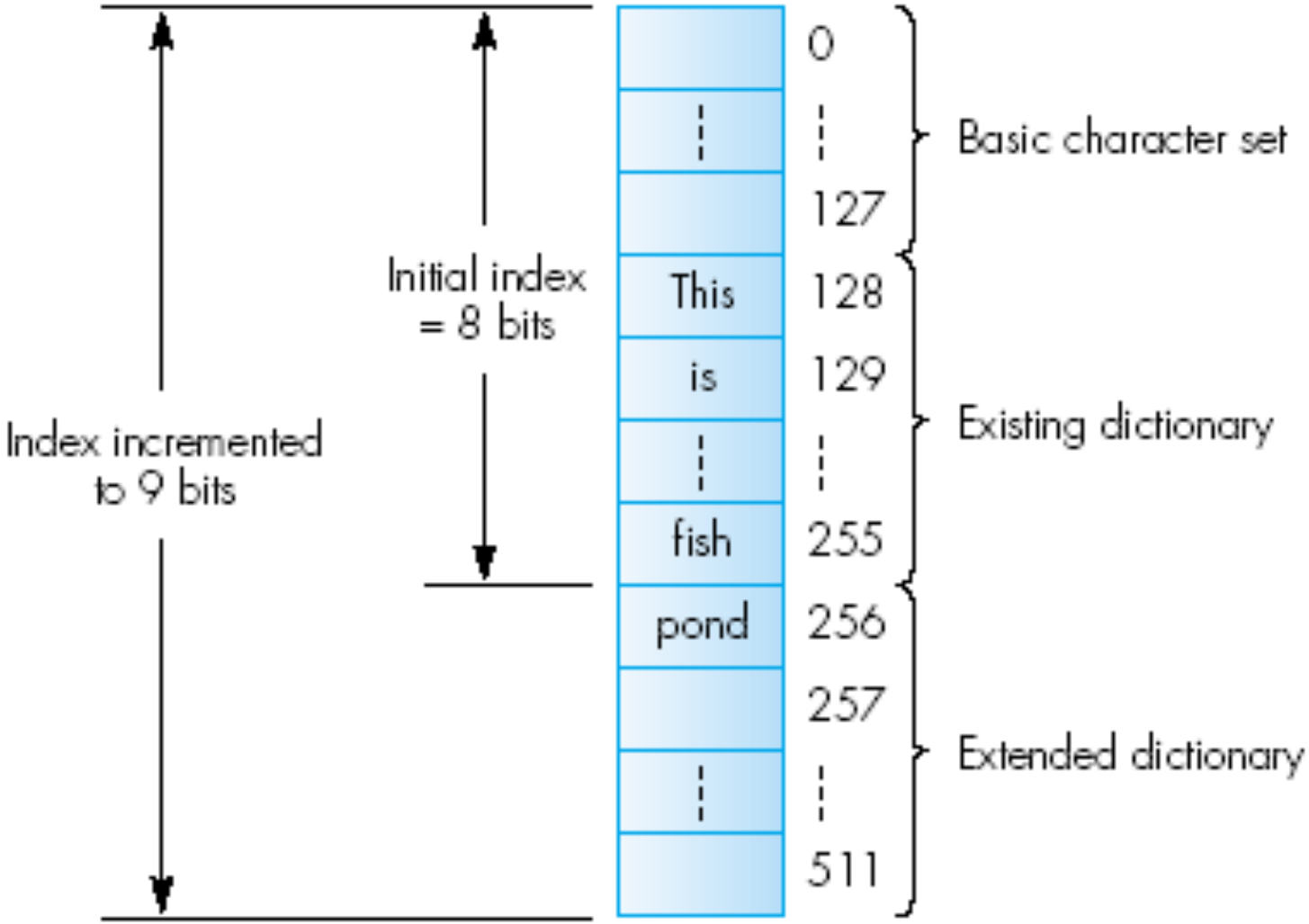
Codificação de Lempel-Ziv-Welch (LZW)

- ✓ **Codificador e decodificador constroem o conteúdo do dicionário dinamicamente enquanto o texto é processado**
- ✓ **Inicialmente, o dicionário contém somente o conjunto de caracteres que foi usado na construção do texto (ex. ASCII 7 bits)**
- ✓ **As entradas restantes são utilizadas para codificar palavras que ocorrem no texto**
- ✓ **Exemplo:**
  - ***This is simple as it is ...***

Exemplo de Lempel-Ziv-Welch



Exemplo de Lempel-Ziv-Welch



Algoritmo LZ77

- **Variação do LZW usado em vários compactadores comerciais**
  - ***PKZIP/PKUNZIP (para o DOS)***
  - ***ARJ***
  - ***GZIP/GUNZIP***
  - ***Compress/Uncompress (para o UNIX)***
- **Princípio de funcionamento:**
  - ***Em uma sequência de caracteres, o algoritmo procura na janela corrente a maior sub-sequência que casa com o início do lookahead buffer e dá como saída um ponteiro para o início da sub-sequência na janela e o primeiro caracter no lookahead buffer que não casa com a sequência. Se não houver sub-sequência, a saída é um ponteiro nulo e o caracter na posição atual da sequência original.***

Algoritmo LZ77

- ✓ **Termos utilizados no algoritmo:**
  - ***Sequência de entrada***
    - Sequência de caracteres a ser comprimida
  - ***Caracter:***
    - Elemento básico da sequência de entrada
  - ***Posição atual:***
    - Posição do caracter na sequência de entrada que está sendo codificado no momento (início do lookahead buffer)
  - ***Lookahead buffer:***
    - Sequência de caracteres a partir da posição atual até o fim da sequência de entrada

Algoritmo LZ77

- ✓ **Termos utilizados no algoritmo:**
  - ***Janela:***
    - Janela de tamanho W contém W caracteres a partir da posição atual em direção ao início da sequência de entrada, i.e. os últimos W caracteres processados
  - ***Ponteiro:***
    - Um ponteiro aponta para a sequência que casa na janela e também indica seu comprimento (L)

Algoritmo LZ77

- ✓ **Algoritmo:**
  1. ***Faça posição atual igual ao início da sequência de entrada***
  2. ***Encontre a maior sub-sequência (longest match) na janela que casa como conteúdo do lookahead buffer***
  3. ***Dê como saída o par (P,C), onde:***
     - P é o ponteiro para a sub-sequência na janela
     - C é o primeiro caracter no lookahead buffer que não casa com a sub-seqüência
  4. ***Se o lookahead buffer não estiver vazio, mova a posição atual (e a janela) L+1 caracteres para frente e volte para o passo 2 (L é o comprimento da sub-sequência).***

Algoritmo LZ77

- ✓ **Exemplo: sequência AABCBBABC**

Pos	1	2	3	4	5	6	7	8	9
Char	A	A	B	C	B	B	A	B	C

- ✓ **Processo de Codificação**

W (janela)

A
AAB
AABC
AABCBB
AABCBBABC

Step	Pos	Match	Char	Saída
1	1	--	A	(0,0) A
2	2	A	B	(1,1) B
3	4	--	C	(0,0) C
4	5	B	B	(2,1) B
5	7	AB	C	(5,2) C

Algoritmo LZ77

- **Decodificação:**
  - *A janela é mantida da mesma forma que na codificação.*
  - *Em cada passo, o algoritmo lê o par (P,C) da entrada e dá como saída a sequência da janela a partir de P seguida do caracter C*
- **Exemplo: (0,0)A (1,1)B (0,0)C (2,1)B (5,2)C**
  1. *(0,0)A => W = nula, saída = A*
  2. *(1,1)B => W = A, saída = AAB*
  3. *(0,0)C => W = AAB, saída = AABC*
  4. *(2,1)B => W = AABC, saída = AABCBB*
  5. *(5,2)C = > W = AABCBB, saída = AABCBBABC*

Algoritmo LZ77

- ✓ **Considerações principais:**
  - ***Boa taxa de compressão para vários tipos de dados***
  - ***Codificação pode ser lenta, já que várias comparações são feitas entre o lookahead buffer a a janela***
  - ***Decodificação muito simples e rápida***
  - ***Requisitos de memória são poucos tanto para codificação quanto para decodificação***
    - A única estrutura mantida em memória é a janela, que normalmente tem tamanho entre 4 e 64 KB.