

Codificação baseada em dicionário

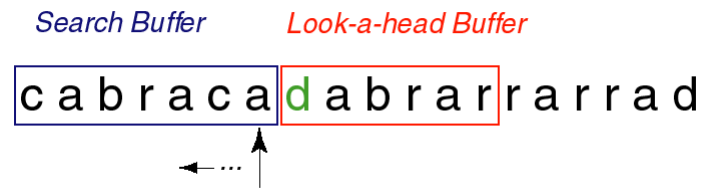
- A codificação baseada em dicionário mais simples, é codificar cada palavra apenas com o número da página do dicionário e a ordem da palavra na folha.
 - Codificação estática
 - Desvantagem: o dicionário tem de ser transmitido
- Dicionários
 - Estáticos
 - Dinâmicos
- Vantagem: Não precisa de conhecer a estatística dos dados.
- Como funciona:
 - Substitui sequências de símbolos por referências ao dicionário.

Codificação baseada em dicionário

- Algoritmo LZ77
 - A sequência de símbolos a codificar é substituída por uma referência a ocorrências anteriores;
 - O dicionário é construído no compressor e no descompressor;
 - Código tem comprimento constante.
- Como funciona:
 - O codificador faz deslizar uma janela com duas partes *Search Buffer* e o *Look-a-head Buffer* (ambos de tamanho fixo);
 - O objectivo é codificar o *Look-a-head Buffer*;
 - O dicionário é o *Search Buffer* (porção dos dados já codificada)
 - Deslocar o *Search Pointer* para encontrar um símbolo igual ao primeiro símbolo do *Look-a-head Buffer*;
 - Contar o número de símbolos iguais nos dois buffers;
 - Gerar o trio <Offset, Length Match, Code(next simbol)> cujo o *Length Match* é maior;
 - Deslocar a janela.

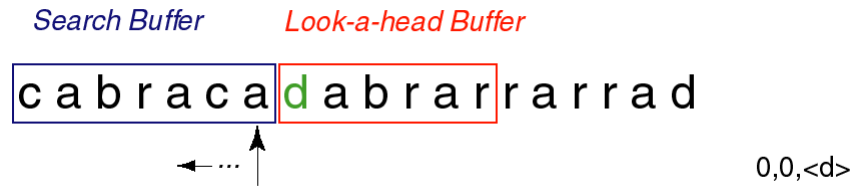
Algoritmo de Compressão LZ77

- Supondo que a dado momento já se tinha codificado o *Search Buffer* na seguinte situação:



Algoritmo de Compressão LZ77

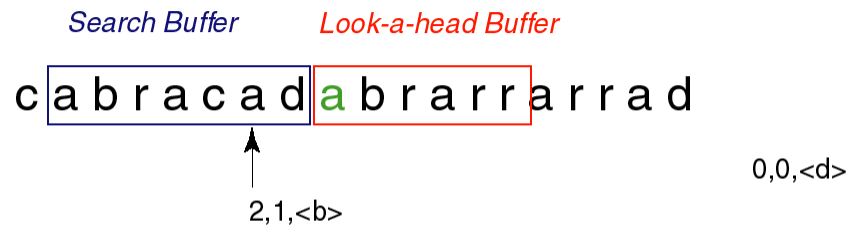
- Supondo que a dado momento já se tinha codificado o *Search Buffer* na seguinte situação:



- Não encontrando nenhum simbolo igual, transmite-se 0,0,<d>

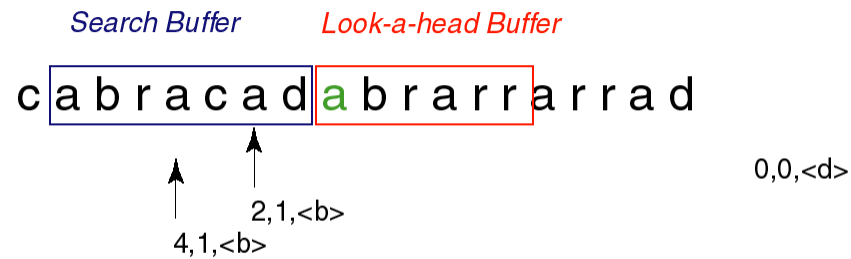
Algoritmo de Compressão LZ77

- Supondo que a dado momento já se tinha codificado o *Search Buffer* na seguinte situação:



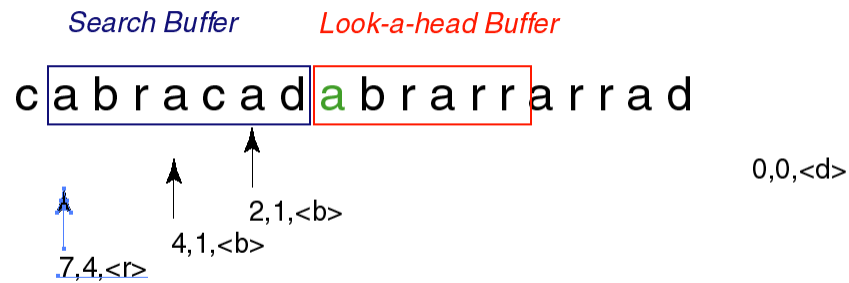
Algoritmo de Compressão LZ77

- Supondo que a dado momento já se tinha codificado o *Search Buffer* na seguinte situação:



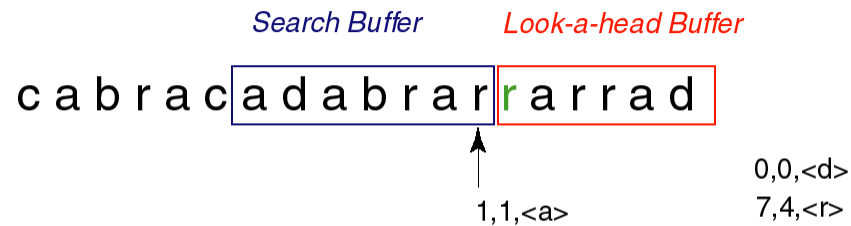
Algoritmo de Compressão LZ77

- Supondo que a dado momento já se tinha codificado o *Search Buffer* na seguinte situação:



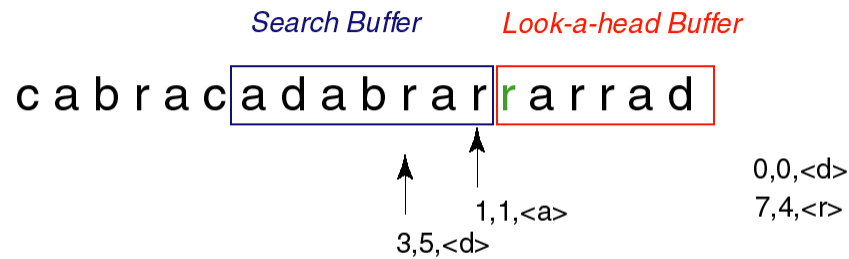
Algoritmo de Compressão LZ77

- Supondo que a dado momento já se tinha codificado o *Search Buffer* na seguinte situação:



Algoritmo de Compressão LZ77

- Supondo que a dado momento já se tinha codificado o *Search Buffer* na seguinte situação:



- Notar que a trama igual começa no *Search Buffer* e prossegue pelo *Look-a-head Buffer*.

Algoritmo de Compressão LZ77

- Supondo que a dado momento já se tinha codificado o *Search Buffer* na seguinte situação:

Search Buffer *Look-a-head Buffer*

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

0,0,<d>
7,4,<r>
3,5,<d>

Algoritmo de Descompressão LZ77

- Supondo que a dado momento já se tinha decodificado o seguinte:

c a b r a c a

0,0,<d>

7,4,<r>

3,5,<d>

Algoritmo de Descompressão LZ77

- Supondo que a dado momento já se tinha decodificado o seguinte:

c a b r a c a d

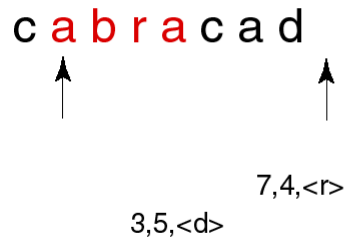
↑
0,0,<d>

7,4,<r>
3,5,<d>

Algoritmo de Descompressão LZ77

- Supondo que a dado momento já se tinha decodificado o seguinte:

c a b r a c a d



The diagram illustrates the state of an LZ77 decoder. The string 'c a b r a c a d' is shown. The characters 'a', 'b', 'r', and 'a' are highlighted in red. Two upward-pointing arrows are positioned below the string: one under the first 'a' and another under the 'd'. Below the first arrow is the label '3,5,<d>'. Below the second arrow is the label '7,4,<r>'. These labels represent the current window pointers (offset and length) and the next character to be decoded.

Algoritmo de Descompressão LZ77

- Supondo que a dado momento já se tinha decodificado o seguinte:

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

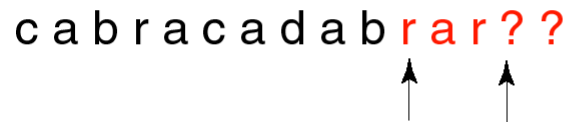
3,5,<d>

7,4,<r>

Algoritmo de Descompressão LZ77

- Supondo que a dado momento já se tinha decodificado o seguinte:

c a b r a c a d a b r a r ? ?

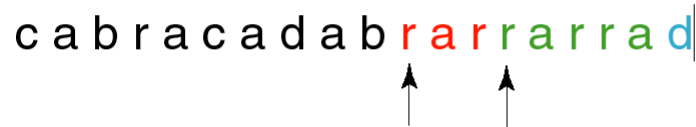


3,5,<d>

Algoritmo de Descompressão LZ77

- Supondo que a dado momento já se tinha decodificado o seguinte:

c a b r a c a d a b **r** **a** **r** **a** **r** **a** **d** |



3,5,<d>

Algoritmo LZ77: Variantes

- Otimização dos trios <offset, length match, code(next simbol)> (codificar com tamanho fixo ou variável) ;
- Usar flag's para evitar a situação <0,0,code>;
- Tamanho dos buffers ser adaptativo;
- Desvantagem:

Search Buffer

Look-a-head Buffer

a b c d e f g h a b c d e f g h a b c

Codificação baseada em dicionário

- Algoritmo Lempel-Ziv-Welch (LZW)
 - Descende do LZ78
 - Compressão adaptativa;
 - Precisa de um dicionário inicial;
 - Dicionário é construído no codificador e no decodificador;
 - O código de cada símbolo é constante
 - Cada código representa streams (de símbolos) com comprimento variável
 - Usado nos formato GIF

Algoritmo de compressão LZW

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

- O LZW original tinha um dicionário com 4096 entradas cujas primeiras 256 eram o código ASCII

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

w = read simbol

while (not end)

 s = read simbol

 if [w s] exists in dictionary

 w = [w s]

 else

 write code for w

 add string [w s] to dictionary with a new code

 w = s

 end

end

write code for w

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “A^BABBABCABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	^B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1		

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “AB^ABBABCABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	^A			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABA**B**ABCBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABAB^BABCABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	^B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABB^ABCABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	^A			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBAB**C**ABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBAB**C**ABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBAB**C**ABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABC^ABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	^A			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCA~~B~~BABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCAB**B**ABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCAB**B**ABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCAB^BABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	^B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABB^ABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	^A			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBA**B**BAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBAB**BAC**”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBAB^BAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	^B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```

w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
    
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			
ABBA				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBA**C**”

```
w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			
ABBA	C			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```

w = read simbol
while ( not end)
    s = read simbol
    if [w s] exists in dictionary
        w = [w s]
    else
        write code for w
        add string [w s] to dictionary with a new code
        w = s
    end
end
write code for w
    
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			
ABBA	C	10	ABBAC	11
C				

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			
ABBA	C	10	ABBAC	11
C	eof			

Algoritmo Lempel-Ziv-Welch

■ mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
end
write code for w
```

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			
ABBA	C	10	ABBAC	11
C		3		

Algoritmo Lempel-Ziv-Welch

- mensagem: “ABABBABCABBABBAC”

```
w = read simbol
while ( not end)
  s = read simbol
  if [w s] exists in dictionary
    w = [w s]
  else
    write code for w
    add string [w s] to dictionary with a new code
    w = s
  end
end
end
write code for s
```

- Código enviado: “1 2 4 5 2 3 6 10 3”

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			
ABBA	C	10	ABBAC	11
C		3		

Algoritmo Lempel-Ziv-Welch

- mensagem: “ABABBABCABBABBAC”
- Assumindo apenas caracteres ASCII a mensagem original tem:

- 16 símbolos x 8bits = 128 bits

- Para a o código transmitido dado que se excedeu os 256 símbolos da tabela ASCII, tenho de enviar mais que 8 bits por cada símbolo

0	0	0	0	x	x	x	x	x	x	x	x
---	---	---	---	---	---	---	---	---	---	---	---

- Com 12 bit o dicionário pode ter 4096 entradas.
- Assim envio 9 códigos x 12 bit = 108 bits.

			Dictionary	
string w	symbol s	output	string	code
			A	1
			B	2
			C	3
A	B	1	AB	4
B	A	2	BA	5
A	B			
AB	B	4	ABB	6
B	A			
BA	B	5	BAB	7
B	C	2	BC	8
C	A	3	CA	9
A	B			
AB	B			
ABB	A	6	ABBA	10
A	B			
AB	B			
ABB	A			
ABBA	C	10	ABBAC	11
C		3		

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

Dictionary				
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3

■ Código enviado: "1 2 4 5 2 3 6 10 3"

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

Dictionary				
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1				

- Código enviado: “1 2 4 5 2 3 6 10 3”
- Mensagem:

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

Dictionary				
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1				

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: A

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: A

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2				

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: A

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
  c = read code
  s = simbol(c)
  if s=[]
    s = w + w[0]
  end
  write s
  add string "w s[0]" to dictionary
  w = s
end
```

Dictionary				
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B			

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: A

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: AB

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B			

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

Dictionary				
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B		AB	4

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: AB

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: AB

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: AB

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB			

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABAB

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB		BA	5

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABAB

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA			

- Código enviado: “1 2 4 5 2 3 6 10 3”
- Mensagem: ABAB

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBA

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA		ABB	6

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBA

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B			

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBA

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B		BAB	7

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBAB

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBAB

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C			

- Código enviado: “1 2 4 5 2 3 6 10 3”
- Mensagem: ABABBAB

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C		BC	8

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBABC

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	C	BC	8

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBABC

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBABC

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	C	BC	8
6	ABB			

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBABCABB

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	C	BC	8
6	ABB		CA	9

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

- Código enviado: "1 2 4 5 2 3 6 10 3"
- Mensagem: ABABBABCABB

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	C	BC	8
6	ABB	ABB	CA	9

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

■ Código enviado: “1 2 4 5 2 3 6 10 3”

■ Mensagem: ABABBABCABB

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	B	BC	8
6	ABB	ABB	CA	9
10	[]			

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

■ Código enviado: “1 2 4 5 2 3 6 10 3”

■ Mensagem: ABABBABCABB

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	B	BC	8
6	ABB	ABB	CA	9
10	ABBA			

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

■ Código enviado: “1 2 4 5 2 3 6 10 3”

■ Mensagem: ABABBABCABBABBA

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	B	BC	8
6	ABB	ABB	CA	9
10	ABBA		ABBA	10

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

■ Código enviado: “1 2 4 5 2 3 6 10 3”

■ Mensagem: ABABBABCABBABBA

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	B	BC	8
6	ABB	ABB	CA	9
10	ABBA	ABBA	ABBA	10

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

■ Código enviado: “1 2 4 5 2 3 6 10 3”

■ Mensagem: ABABBABCABBABBA

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	B	BC	8
6	ABB	ABB	CA	9
10	ABBA	ABBA	ABBA	10
3	C			

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

■ Código enviado: “1 2 4 5 2 3 6 10 3”

■ Mensagem: ABABBABCABBABBAC

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	B	BC	8
6	ABB	ABB	CA	9
10	ABBA	ABBA	ABBA	10
3	C		ABBAC	11

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

■ Código enviado: “1 2 4 5 2 3 6 10 3”

■ Mensagem: ABABBABCABBABBAC

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	B	BC	8
6	ABB	ABB	CA	9
10	ABBA	ABBA	ABBA	10
3	C	C	ABBAC	11

Algoritmo de decompressão LZW

```
c = read code
write simbol(c)
w = simbol(c)
while ( not end )
    c = read code
    s = simbol(c)
    if s=[]
        s = w + w[0]
    end
    write s
    add string "w s[0]" to dictionary
    w = s
end
```

■ Código enviado: “1 2 4 5 2 3 6 10 3”

■ Mensagem: ABABBABCABBABBAC

			Dictionary	
input	s	w	w + s[0]	code
			A	1
			B	2
			C	3
1		A		
2	B	B	AB	4
4	AB	AB	BA	5
5	BA	BA	ABB	6
2	B	B	BAB	7
3	C	B	BC	8
6	ABB	ABB	CA	9
10	ABBA	ABBA	ABBA	10
3	C	C	ABBAC	11