

# Codifica di Huffman e Lempel-Ziv-Welch

1

ALBERTO BELUSSI  
ANNO ACCADEMICO 2010/2011

# Tipi di compressione



- **Senza perdita (lossless)**: permettono di ricostruire perfettamente la rappresentazione del dato originale, conservano tutta l'informazione (esempio, Huffman e Lempel-Ziv-Welch)
- **Con perdita (lossy)**: permettono di ricostruire solo in parte la rappresentazione del dato originale, quindi parte dell'informazione va persa.
  - Le tecniche lossy si dividono in:
    - Tecniche basate sulla predizione
    - **Tecniche orientate alla frequenza** (*Discrete Cosine Transform - DCT*; usata ad esempio nella codifica JPEG)
    - Tecniche orientate all'importanza

# Codifica di Huffman

3

Note le caratteristiche della sorgente:

- Ad esempio

- Alfabeto={A,B,C,D}

- $P(A)=0,50$

$$I(A)=-\log_2(0,50)=1 \text{ bit}$$

- $P(B)=0,20$

$$I(B)=-\log_2(0,20)=2,322 \text{ bit}$$

- $P(C)=0,18$

$$I(C)=-\log_2(0,18)=2,474 \text{ bit}$$

- $P(D)=0,12$

$$I(D)=-\log_2(0,12)=3,059 \text{ bit}$$

$$H_{\text{sorgente}} = 0,5 \times 1 + 0,2 \times 2,322 + 0,18 \times 2,474 + 0,12 \times 3,059 = \mathbf{1,777 \text{ bit/simb}}$$

# Codifica di Huffman

4

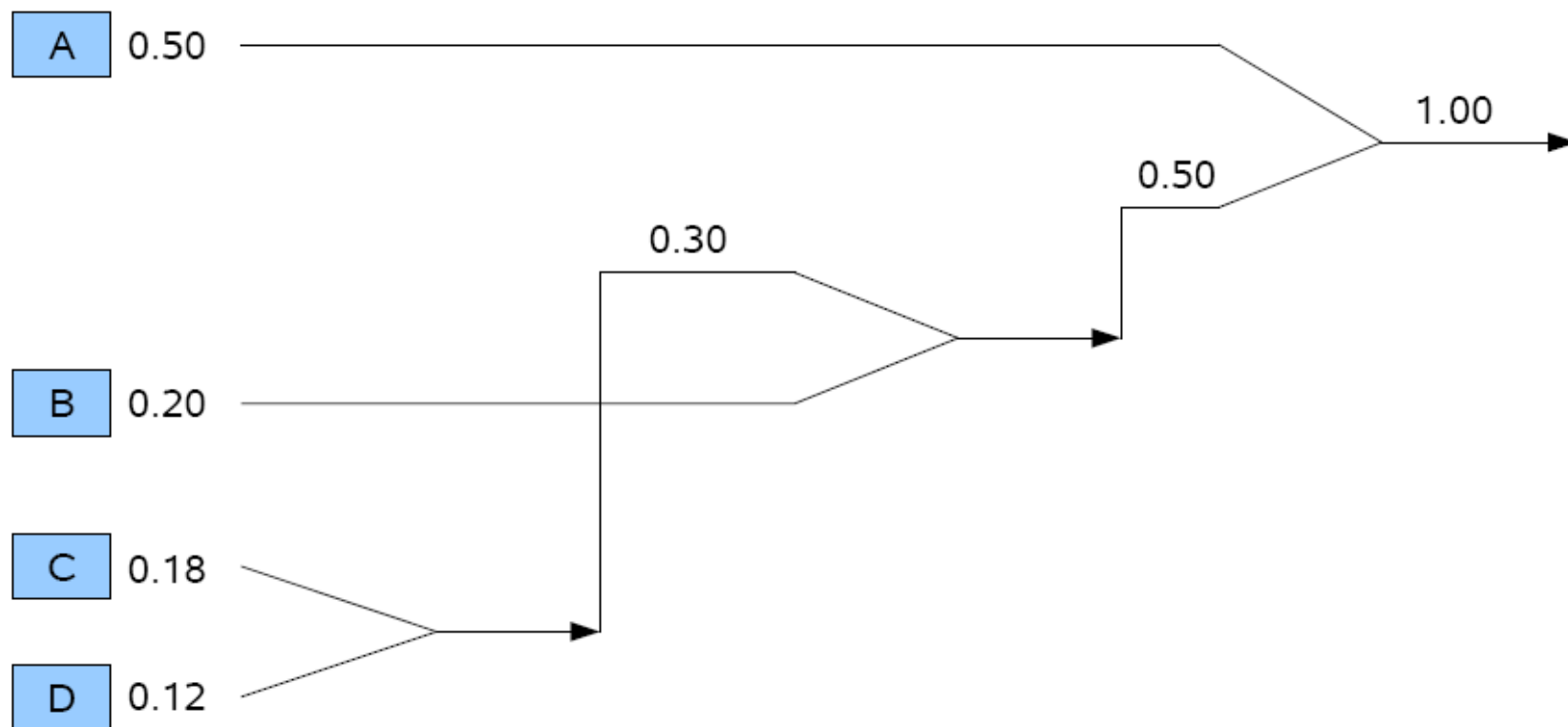
Si applica il seguente algoritmo:

1. Si ordinano i simboli in ordine **decrescente di** probabilità;
2. Si unificano i due simboli in fondo alla lista, si sommano le probabilità e si reintroduce tale probabilità nella lista mantenendo l'ordine;
3. Si ripete il passo precedente fino a quando la lista sarà formata da un solo elemento (di probabilità unitaria);

.... continua →

# Codifica di Huffman

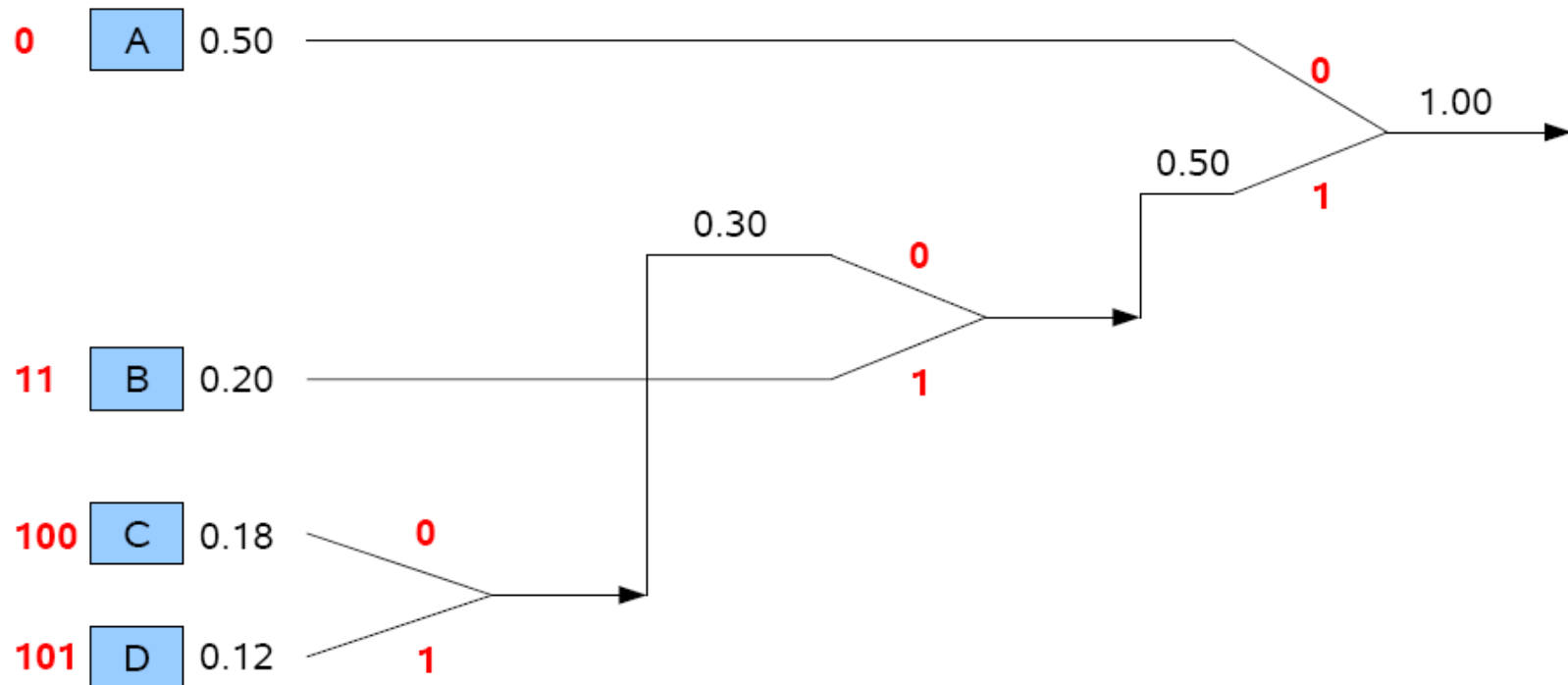
5



# Codifica di Huffman

6

4. Si etichetta ogni biforcazione con uno 0 e un 1;
5. La codifica di ogni simbolo dell'alfabeto è data dal percorso dalla radice dell'albero fino alla foglia contenente il simbolo;
6. Per le proprietà degli alberi, ogni foglia ha un unico percorso (codifica) che non sarà mai il prefisso di un altro percorso (la codifica di un simbolo non può essere l'inizio della codifica di un altro simbolo).



# Codifica di Huffman

8

La codifica è immediata:

- es. AABABCABDACB di 12 simboli, sarà codificato con: 0 0 1 1 0 1 1 1 0 0 0 1 1 1 0 1 0 1 0 0 1 1

il numero totale di bit è 22 (in media  $22/12=1.833$  bit/simbolo); se si fosse codificato senza tener conto delle probabilità (2 bit a simbolo) il messaggio occuperebbe 24 bit.

# Codifica di Huffman

9

La **decodifica** è **univoca e mai ambigua**.

- es. 10010011011000 viene suddivisa senza ambiguità nei simboli
- 100 100 11 0 11 0 0 0
- C C B A B A A A
- Il primo bit di ogni simbolo coincide con la radice dell'albero; ogni bit successivo guida il decodificatore fino ad una foglia (simbolo corrispondente).

# Codifica di Huffman

10

$p_k$	$I(s_k)$	$p_k \cdot I(s_k)$	$L_k$	$L_k \cdot p_k$
0,500	1,000	0,500	1,000	0,500
0,200	2,322	0,464	2,000	0,400
0,180	2,474	0,445	3,000	0,540
0,120	3,059	0,367	3,000	0,360
1,000		<b>1,777</b>		<b>1,800</b>

Entropia: 1,777  
bit/simbolo

Lunghezza media del  
codice (binit/simbolo)

Un messaggio di es. 1000 simboli avrà lunghezza media di 1800  
binit (il minimo teorico è 1777)

E' possibile **avvicinarsi quanto si vuole all'entropia**, codificando coppie, terne  
ecc. di simboli dell'alfabeto (ciascuna con la sua probabilità)

# Codifica LZW

11

## Confronto con codifica Huffman

- Nella codifica di *Huffman* ogni **simbolo** ha una codifica a lunghezza variabile: più il simbolo ha probabilità bassa più la sua codifica è lunga in termini di bit.
- Nella codifica *LZW* si codificano **sequenze di simboli di lunghezza variabile** con codici di lunghezza fissa
- Nella codifica *Huffman* va trasmessa anche la tabella di codifica, nella codifica *LZW* non è necessario (la tabella viene ricostruita durante la decodifica!).

# Codifica LZW

12

## Algoritmo di codifica

- Si inizializza il dizionario dei simboli con tutti i simboli dell'alfabeto sorgente con la corrispondente codifica
- Si inizializza a “” la variabile PREFISSO
- Si legge il primo carattere dello STREAM di INPUT nella variabile C.



Seq. simboli	code
A	0000
B	0001
C	0010

# Codifica LZW

13

## Algoritmo di codifica

- Se la sequenza Prefisso+C è presente nel dizionario
  - allora  $\text{Prefisso} := \text{Prefisso} + C$ ;
  - altrimenti
    - ✦  $\text{output} := \text{output} + \text{code}(\text{Prefisso})$
    - ✦ inserire la sequenza Prefisso+C nel dizionario e codificarla
    - ✦  $\text{Prefisso} := C$
- Se esistono ancora caratteri in INPUT leggere un carattere e tornare al punto precedente altrimenti:
  - ✦  $\text{output} := \text{output} + \text{code}(\text{Prefisso})$



Stringa di simboli	code
A	0000
B	0001
C	0010

# Codifica LZW

14

Algoritmo di codifica: esempio



	prefisso	“”
input	ABABCBABAB	
output		

Stringa di simboli	code
A	0000
B	0001
C	0010



	prefisso	<del>A</del> B
input	ABABCBABAB	
output	0000	

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011

1

2

3

# Codifica LZW

15

Algoritmo di codifica: esempio

	<div>↓</div> <div>prefisso</div> <div><del>B</del>A</div>
input	ABABCBABAB
output	0000 0001

	<div>↓</div> <div>prefisso</div> <div>AB</div>
input	ABABCBABAB
output	0000 0001

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011
BA	0100

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011
BA	0100

# Codifica LZW

16

Algoritmo di codifica: esempio


		prefisso	A <del>B</del> C
input	ABABCBABAB		
output	0000 0001 0011		

Diagram illustrating the LZW encoding process. A red arrow points down from the title to the input string. The input string is ABABCBABAB. The output string is 0000 0001 0011. The prefix is A, and the current symbol is B. The next symbol is C. The output string is 0000 0001 0011. The next symbol is C. The output string is 0000 0001 0011.

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011
BA	0100
ABC	0101

A questo punto poiché “ABC” non è presente nel dizionario allora codifico AB, inserisco nel dizionario ABC e metto nella variabile *prefisso* C. Procedendo così ottengo alla fine:

output	0000 0001 0011 0010 0100 0111
--------	-------------------------------

# Decodifica LZW

17

## Algoritmo di decodifica

- Si inizializza il dizionario dei simboli con tutti i simboli dell'alfabeto sorgente ordinati e si codificano in stringhe di bit
- Si inizializza a “” la variabile PREFISSO
- Si legge la prima posizione della codifica dello STREAM di INPUT nella variabile CW.
- Si supponga che esista una funzione DECODE(c) che dato un codice c restituisce la corrispondente stringa di simboli del dizionario.

PW CW  
↓ ↓

prefisso	“”
----------	----

input	0000 0001 0011 0010 0100 0101
output	

Stringa di simboli	code
A	0000
B	0001
C	0010



# Decodifica LZW

18



1. Si produce in output la stringa di simboli  $\text{DECODE}(\text{CW})$
2.  $\text{PW} := \text{CW}$ ; sposto CW sulla posizione successiva
3. Se CW è presente nel dizionario allora
  - ✦ Si produce in output la stringa di simboli  $\text{DECODE}(\text{CW})$
  - ✦  $\text{Prefisso} := \text{DECODE}(\text{PW})$
  - ✦  $\text{Char} := \text{DECODE}(\text{CW})[1]$  (primo carattere della stringa decodificata)
  - ✦ Si aggiunge la stringa  $\text{Prefisso} + \text{Char}$  nel dizionario
4. Altrimenti
  - ✦  $\text{Prefisso} := \text{DECODE}(\text{PW})$
  - ✦  $\text{Char} := \text{DECODE}(\text{PW})[1]$  (primo carattere della stringa decodificata al passo precedente)
  - ✦ Si produce in output  $\text{Prefisso} + \text{Char}$  e si aggiunge al dizionario
5. Se l'input non è terminato tornare al punto 2.

# Decodifica LZW

19

	PW	CW			
					
input	0000 0001 0011 0010 0100 0111				
output	A				

prefisso	“”
----------	----

	PW	CW	
			
input	0000 0001 0011 0010 0100 0111		
output	A <b>B</b>		
prefisso	A		char B

Stringa di simboli	code
A	0000
B	0001
C	0010

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011

# Decodifica LZW

20

		PW	CW	
		↓	↓	
input	0000 0001 0011 0010 0100 0111			
output	ABAB			
prefisso	B		char	A

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011
BA	0100

# Decodifica LZW

21

		PW	CW
		↓	↓
input	0000 0001 0011 0010 0100 0111		
output	ABABC		
prefisso	AB		char C

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011
BA	0100
ABC	0101

# Decodifica LZW

22

				PW	CW
				↓	↓
input	0000 0001 0011 0010 0100 0111				
output	ABABCBA				
prefisso	C			char	B

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011
BA	0100
ABC	0101
CB	0110

# Decodifica LZW

23

input	0000 0001 0011 0010 0100 0111					
output	ABABCBA <b>BAB</b>					
prefisso	BA		char	B		

PW  
↓

CW  
↓

Stringa di simboli	code
A	0000
B	0001
C	0010
AB	0011
BA	0100
ABC	0101
CB	0110

0111???