

# **L'informazione e la sua codifica**

**Corso di**

**Introduzione all'Informatica per l'Editoria**

**Roberto Maieli - Roma Tre**

# **Informazione e Informatica**

# Informatica e telecomunicazione

## ➤ Cos'è l'informatica?

- lo studio sistematico degli algoritmi che descrivono e trasformano l'**informazione**: la loro teoria, analisi, progetto, efficienza, realizzazione e applicazione  
[ACM - Association for Computing Machinery]
- **la scienza della rappresentazione e dell'elaborazione dell'informazione**

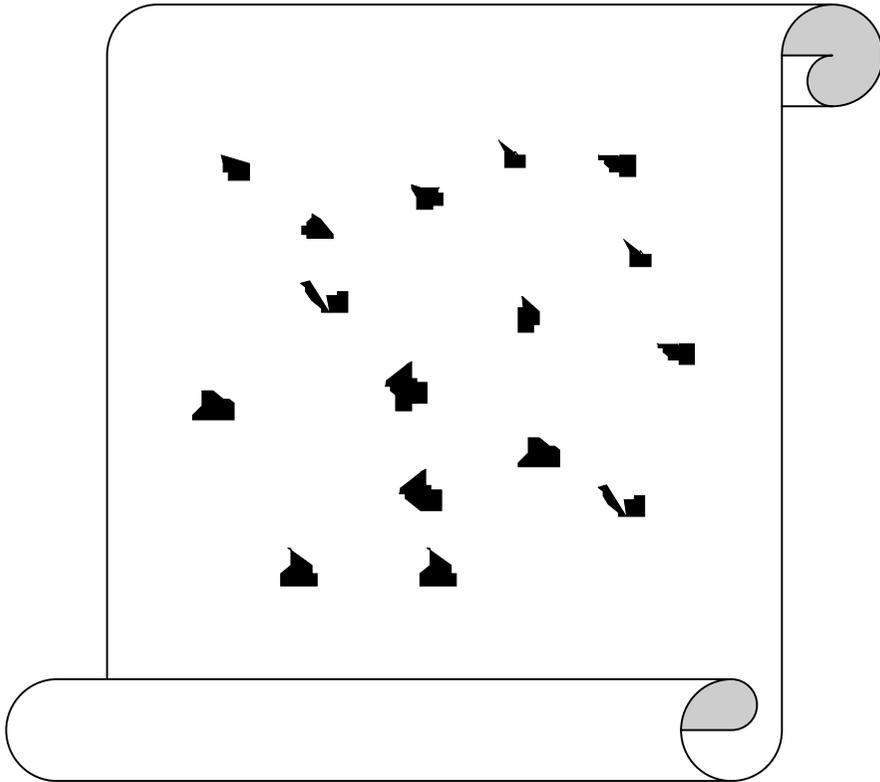
## ➤ Cos'è la telecomunicazione?

- la trasmissione **rapida** a **distanza** dell'informazione

## ➤ Attenzione:

- **Non** si parla di tecnologia dei **calcolatori** !
- Si attribuisce ruolo centrale al concetto di **informazione** !

# Il concetto di informazione



**Configurazione 1**



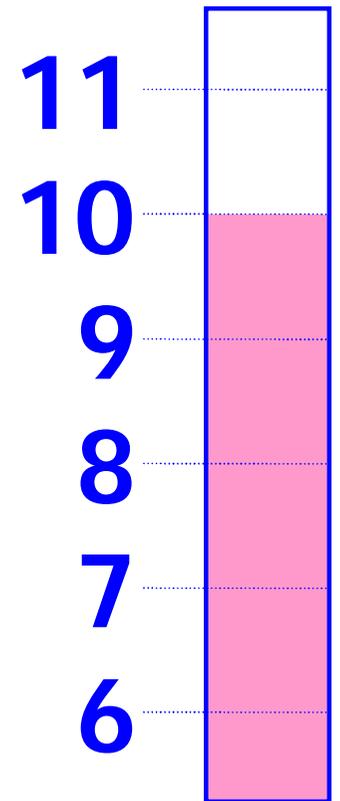
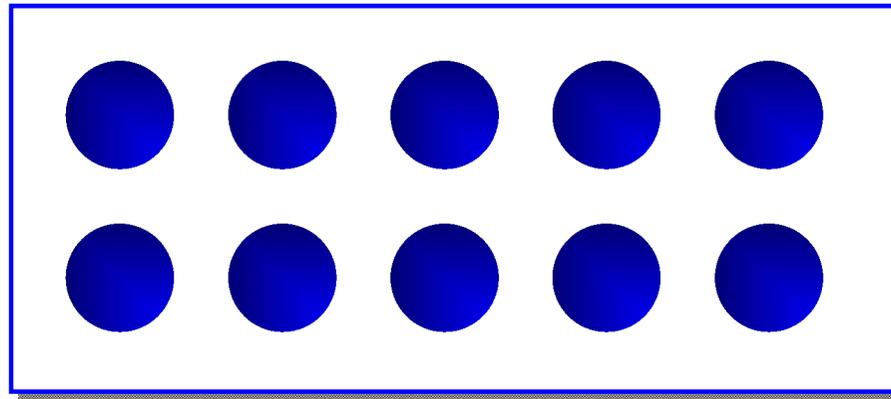
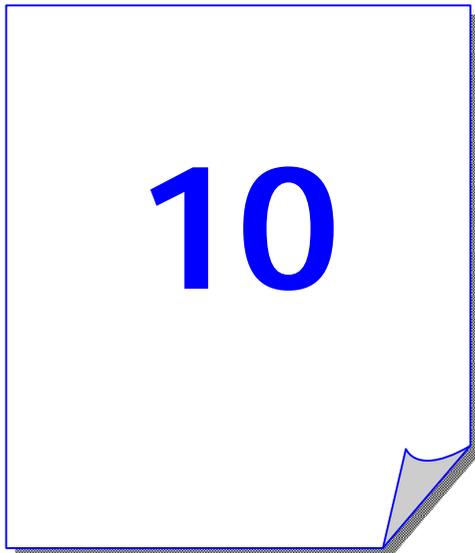
**Configurazione 2**

# Informazione e supporto

- L'informazione è "portata da", o "trasmessa su", o "memorizzata in", o "contenuta in" qualcosa; questo "qualcosa" però non è l'informazione stessa.
- Ogni supporto ha le sue caratteristiche in quanto mezzo su cui può essere scritta dell'informazione.
- **Esempi:** l'aria, un CD sono supporti

# Informazione e supporti

La “**stessa informazione**” può essere scritta su **supporti differenti**.



**Denotano** la stessa informazione ma **significano** forse **cose diverse**

# Informazione e supporti

Lo **stesso supporto** può portare **informazioni differenti**.

**fare**

inglese

to make, to do, to build, ...

tariffa, prezzo, ...

italiano

**burro**

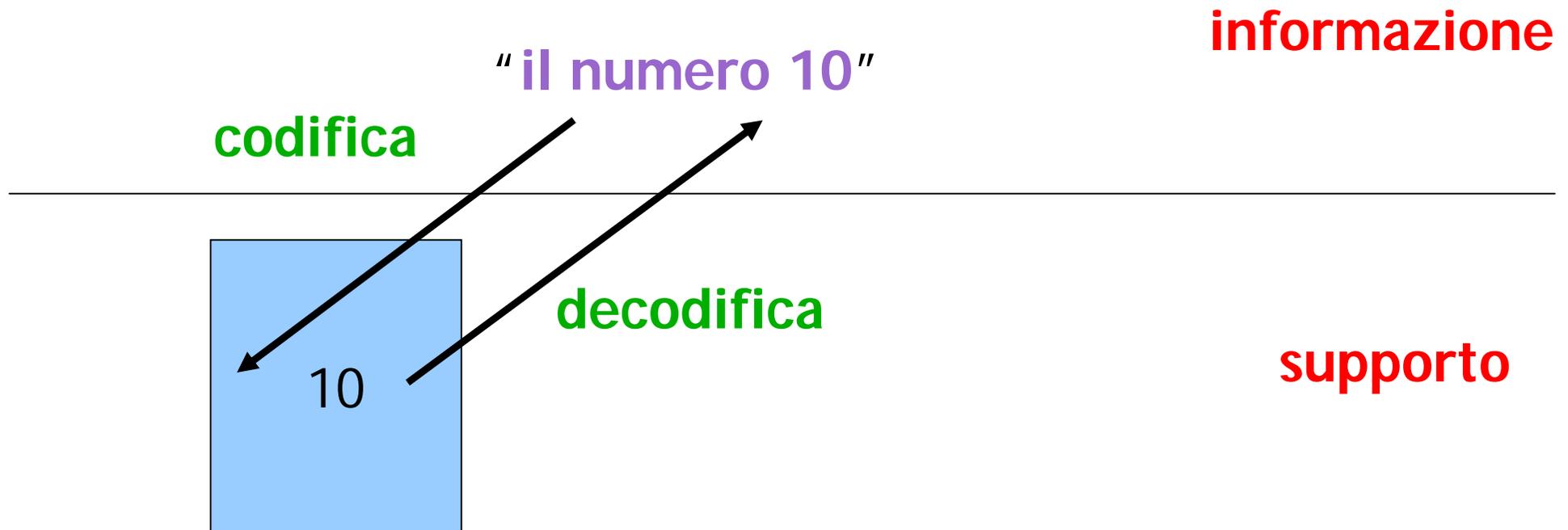
spagnolo

mantequilla, manteca, ...

asino, cavalletto, somaro, ...

# Informazione vs Supporto

- **Codifica:** l'operazione con la quale l'informazione viene **scritta** su un supporto
- **Decodifica:** l'operazione con la quale l'informazione viene **letta** dal supporto



# Informazione e supporto

➤ Distinguere informazione e supporto fisico è distinguere tra

“entità logiche” ed “entità fisiche”:

- L'informazione **richiede un supporto fisico**, ma non coincide con esso;
- L'informazione è un'entità **extra-fisica**, non interpretabile in termini di materia-energia e sottoposta alle leggi della fisica solo perché basata su un supporto fisico.

➤ L'informazione si può **creare e distruggere**.

**Claude Shannon (1916-2001):**

“informazione” = “entropia” (dal greco *trasformazione*)

# Quali caratteristiche deve avere un sistema fisico per supportare informazioni?

- Si ottiene informazione quando, dato un **insieme di alternative possibili**, la lettura del (o l'accesso al) supporto ne esclude alcune e ne seleziona altre.
- **Condizione necessaria** perché un supporto possa portare informazione è che possa assumere **configurazioni differenti**, a ognuna delle quali venga **associata** una differente **entità di informazione**.

# Supporto fisico: 1<sup>a</sup> condizione



- Deve consentire di potere esprimere delle differenze
  - Es: voglio rappresentare 2 alternative



- Cosa **rappresenta** ciascuna configurazione?

# Supporto fisico: 2a condizione

Deve essere condivisa una **regola** per attribuire un **significato** (entità informativa) a ciascuna **configurazione**

Configurazione 1

Configurazione 2



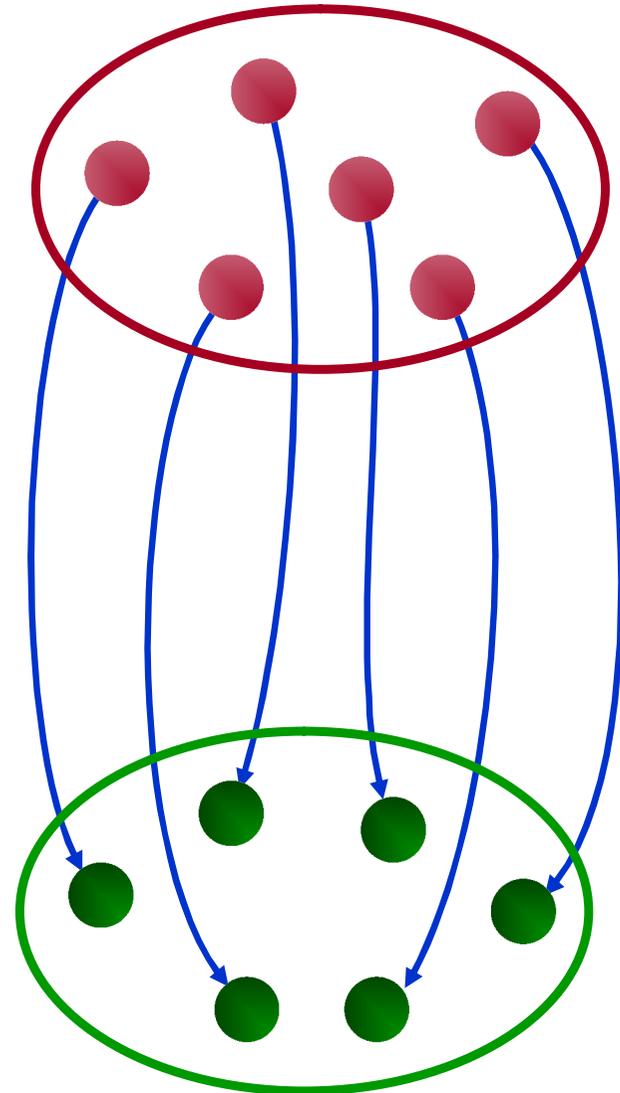
# Definire un codice

## ➤ Identificare

- { Configurazioni }
- { Entità di informazione }

## ➤ Associare gli elementi dei 2 insiemi

**Codice**: è una regola che consente di associare ad ogni **configurazione ammessa** una **entità di informazione**

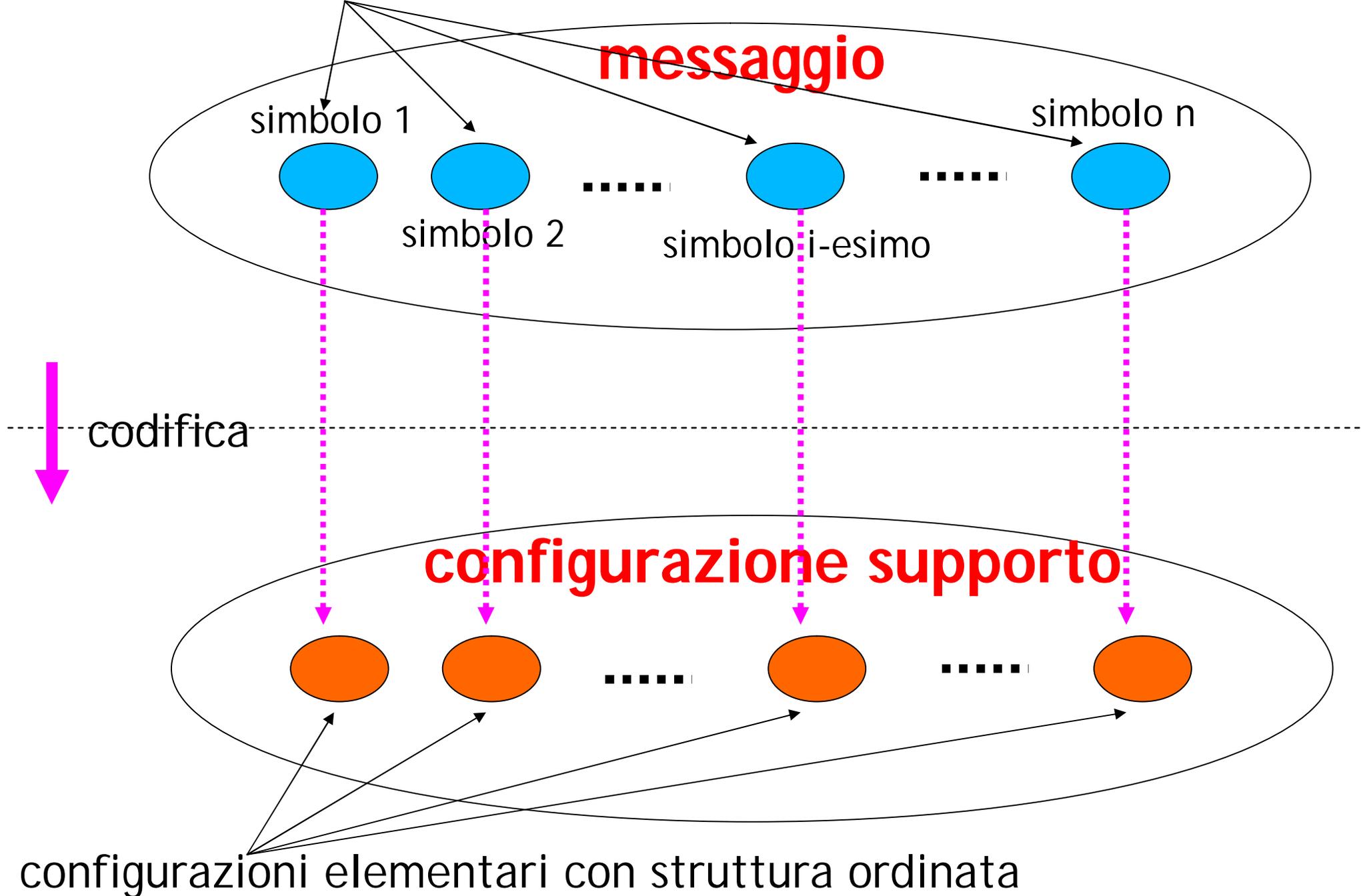


# Configurazioni, configurazioni elementari, simboli e messaggi 1/2

## Principio di composizionalità dell'informazione:

- la **configurazione del supporto** risulta dall'insieme ordinato delle configurazioni elementari
- **l'informazione complessiva** portata dal supporto, detta **messaggio**, si ricava dall'insieme ordinato dei **simboli**, cioè le entità elementari di informazioni corrispondenti alle configurazioni elementari.

entità elementari di informazione con struttura ordinata



# Configurazioni, configurazioni elementari, simboli e messaggi 2/2

## Principio di composizionalità dell'informazione:

volendo leggere un messaggio contenuto in un supporto (es. foglio di carta) dobbiamo disporre di un **codice** che:

**Regola 1:** associ configurazioni elementari (es. macchie sul foglio) a simboli (es. lettere dell'alfabeto)

**Regola 2:** indichi come comporre i simboli per ottenere un messaggio (es. leggere le lettere di ogni riga da sx a dx, dall'alto in basso)

# Tre diversi livelli di informazione

- **Informazione sintattica:** ci si chiede se un certo supporto sia **adatto**, in base alle configurazioni assumibili, a mantenere un certo messaggio
- **Informazione semantica:** ci si chiede quale **significato** sia da attribuire ad una data configurazione del supporto fisico

# Teoria dell'informazione

## (Claude Shannon, 1916-2001)

- Quando si parla di “teoria dell'informazione” si fa riferimento al solo livello “sintattico”
- **Ambito di applicazione:** caratterizzare le condizioni per la trasmissione di segnali in termini di
  - **adeguatezza** del supporto adottato per la trasmissione
  - **accuratezza** della trasmissione stessa.
- **Quali problemi si pone:**
  - un certo supporto può essere utilizzato per la memorizzazione di una certa **quantità di informazione**?
  - con quale **velocità** una certa quantità di informazione può essere trasferita a distanza mediante un certo supporto?
  - con quale grado di **accuratezza** un certo messaggio è stato trasmesso?

# Informazione e incertezza

- La presenza di informazione è condizionata dal fatto che il supporto sia in grado di assumere **diverse configurazioni**.
- Se la nostra ignoranza, o più formalmente la nostra **incertezza**, circa l'effettiva configurazione del supporto viene ridotta dall'accesso al supporto, allora sembra del tutto ragionevole affermare che tale atto ci ha **portato dell'informazione**.
- Se fossimo in grado di misurare il **grado di incertezza** in cui ci trovavamo **prima della lettura e quello successivo a essa**, la **quantità di informazione** portata dalla configurazione che abbiamo letto sul supporto potrebbe essere definita proprio dalla **differenza tra tali gradi di incertezza**.

# Informazione e incertezza

Quanto meno è **probabile** che si presenti una configurazione tanto maggiore è l'informazione che essa porta

(Hartley & Shannon, Teoria Matematica della Comunicazione, 1948)

- La **quantità di informazione** che si ottiene selezionando una configurazione da un insieme che ne contiene 2 costituisce l'**unità elementare di informazione**, chiamata **BIT**
- Analogamente: 1 BIT è la quantità di informazione che può esprimere un supporto che ammette solo 2 configurazioni (0,1)
- La risposta SI oppure NO ad una domanda porta dunque 1 BIT di informazione.

# **La codifica dell'informazione**

# Codifica dati e istruzioni

## ➤ **Algoritmo**

- **descrizione** della **soluzione di problema** scritta in modo da poter essere eseguita da un **esecutore** (eventualmente diverso dall'autore dell'algoritmo)
- sequenza di **istruzioni** che operano su **dati**.

## ➤ **Programma**

- **algoritmo** scritto in modo da poter essere eseguito da un **calcolatore** (esecutore automatico)

- Per scrivere un **programma** è necessario rappresentare **istruzioni** e **dati** in un formato tale che **l'esecutore automatico** sia capace di **memorizzare** e **manipolare**.

# Codifica dati e istruzioni

## ➤ Alfabeto dei simboli

- cifre "0", "1", ..., "9", separatore decimale (","), separatore delle migliaia (".") e segni positivo ("+") o negativo ("-").

## ➤ Regole di composizione (**sintassi**), che definiscono le successioni "ben formate"

- "1.234,5" è la rappresentazione di un numero;
- "1,23,45" non lo è.

## ➤ Codice (**semantica**): interpretazione posizionale

- "1.234,5" =  $1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$
- "1,23,45" = ??

## ➤ Lo stesso alfabeto può essere utilizzato con **codici diversi**:

- "123,456" =  $1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2} + 6 \times 10^{-3}$ , [IT]
- "123,456" =  $1 \times 10^5 + 2 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$ , [UK]

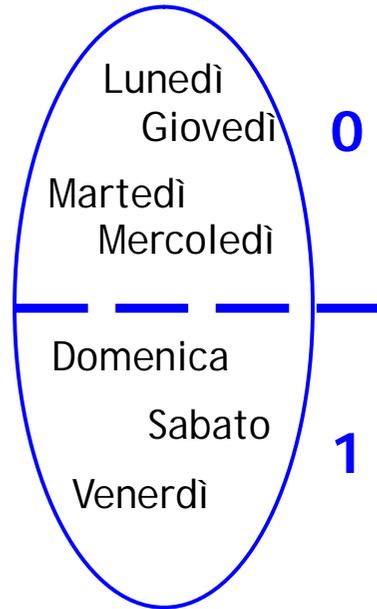
# Codifica Binaria

- **Alfabeto binario:** usiamo dispositivi con solo due stati (**0,1**)
- **Problema:** assegnare un **codice binario univoco** a tutti gli oggetti compresi in un insieme predefinito (e.g. studenti)
- **PB:** Quanti **oggetti** posso codificare con **k bit** ?
  - 1 bit  $\Rightarrow$  2 stati (0, 1)  $\Rightarrow$  2 oggetti (e.g. Vero/Falso)
  - 2 bit  $\Rightarrow$  4 stati (00, 01, 10, 11)  $\Rightarrow$  4 oggetti
  - 3 bit  $\Rightarrow$  8 stati (000, 001, ..., 111)  $\Rightarrow$  8 oggetti
  - ...
  - **k bit  $\Rightarrow$   $2^k$  stati  $\Rightarrow$   $2^k$  oggetti**
- **PB:** Quanti **bit** mi servono per codificare **N** oggetti ?
  - $N \leq 2^k \Rightarrow k \geq \log_2 N \Rightarrow k = \lceil \log_2 N \rceil$  (approssimato all'intero superiore)
- **Attenzione!**  
ipotesi implicita: i codici hanno tutti la **stessa lunghezza**

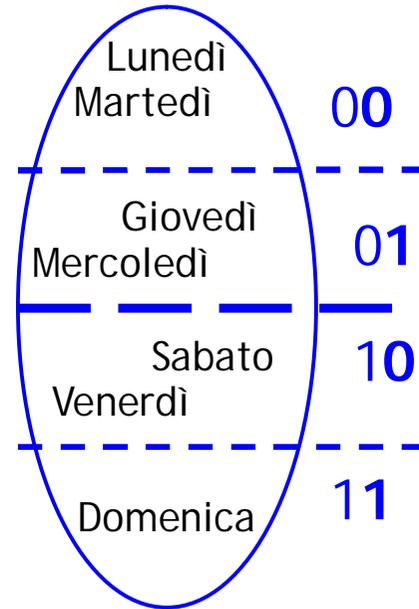
# Esempio di codifica binaria

- **Problema:** assegnare un codice binario univoco a tutti i giorni della settimana
- **Giorni della settimana:**  $N = 7 \Rightarrow k \geq \log_2 7 \Rightarrow k = 3$
- **Con 3 bit possiamo ottenere 8 diverse configurazioni:**
  - Ne servono 7, quali utilizziamo?
  - Quale configurazione associamo a quale giorno?
- **Attenzione:** ipotesi che i codici abbiano tutti la stessa lunghezza

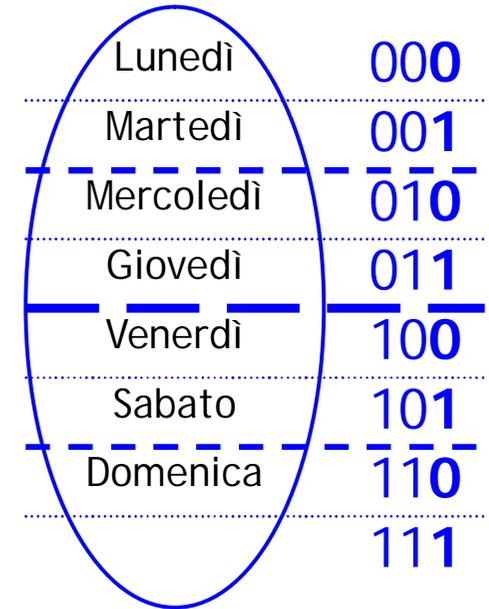
# I giorni della settimana in binario (1)



**1 bit**  
**2 "gruppi"**

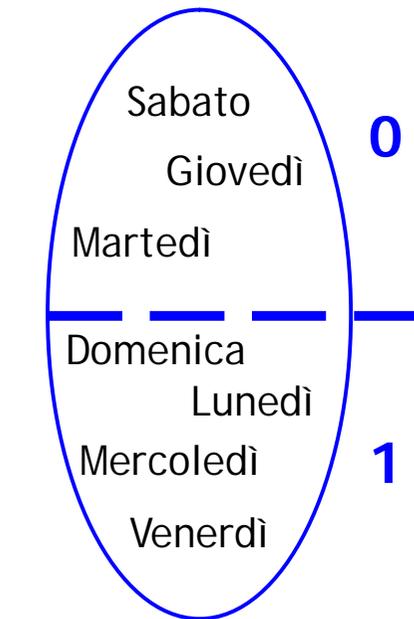


**2 bit**  
**4 "gruppi"**

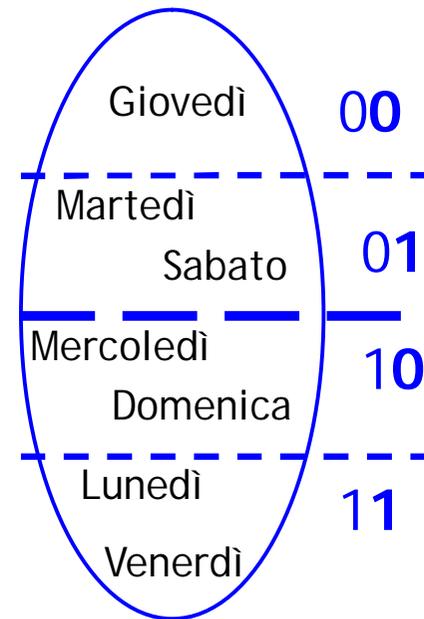


**3 bit**  
**8 "gruppi"**

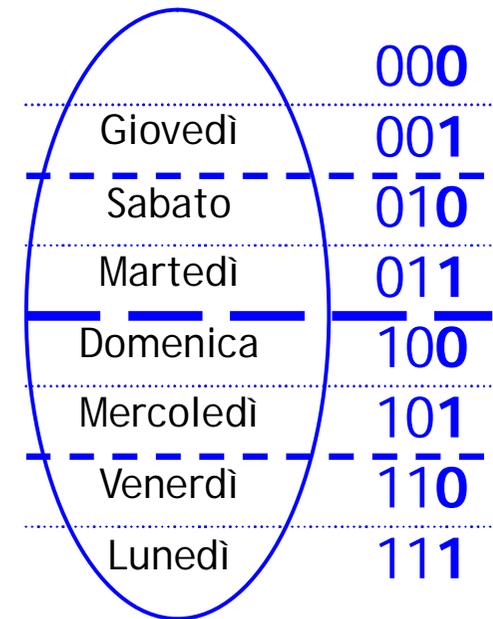
# I giorni della settimana in binario (2)



**1 bit**  
**2 "gruppi"**



**2 bit**  
**4 "gruppi"**



**3 bit**  
**8 "gruppi"**

# Codifica binaria dei caratteri

## ➤ Quanti sono gli oggetti compresi nell'insieme?

- 26 lettere maiuscole + 26 minuscole  $\Rightarrow$  52
- 10 cifre
- Circa 30 segni d'interpunzione
- Circa 30 caratteri di controllo (CANC, ESC, INVIO, CTRL, ALT ...)

**circa 120 oggetti complessivi  $\Rightarrow k = \lceil \log_2 120 \rceil = 7$**

## ➤ Codice ASCII: utilizza 7 bit e quindi può rappresentare al massimo $2^7=128$ caratteri

- Con 8 bit (= byte) rappresento  $256=2^8$  caratteri (ASCII ext.)
- Oggi si usano codici più estesi: **UNICODE**  $2^{16} = 65535$   
per rappresentare anche i caratteri delle lingue orientali

# ASCII su 7 bit

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
010	sp	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	Y	z	{		}	~	canc

# Bit, Byte, KiloByte, MegaByte, ...

**Bit** = solo due stati, "0" oppure "1".

**Byte** = 8 bit, quindi  $2^8 = 256$  stati

**KiloByte [KB]** =  $2^{10}$  Byte = 1024 Byte ~  $10^3$  Byte (diff.2%)

**MegaByte [MB]** =  $2^{20}$  Byte = 1'048'576 Byte ~  $10^6$  Byte

**GigaByte [GB]** =  $2^{30}$  Byte ~  $10^9$  Byte

**TeraByte [TB]** =  $2^{40}$  Byte ~  $10^{12}$  Byte (differenza 10%)

**PetaByte [PB]** =  $2^{50}$  Byte ~  $10^{15}$  Byte

**ExaByte [EB]** =  $2^{60}$  Byte ~  $10^{18}$  Byte

# Lo standard IEC per i prefissi binari

Grandezza	Nome	Simbolo		Dimensione	SI	Diff. %
Kilo binario	Kibi	Ki	$2^{10}$	1'024	$10^3$	2.40%
Mega binario	Mebi	Mi	$(2^{10})^2$	1'048'576	$(10^3)^2$	4.86%
Giga binario	Gibi	Gi	$(2^{10})^3$	1'073'741'824	$(10^3)^3$	7.37%
Tera binario	Tebi	Ti	$(2^{10})^4$	1'099'511'627'776	$(10^3)^4$	9.95%
Peta binario	Pebi	Pi	$(2^{10})^5$	1'125'899'906'842'624	$(10^3)^5$	12.59%
Exa binario	Exbi	Ei	$(2^{10})^6$	1'152'921'504'606'846'976	$(10^3)^6$	15.29%
Zetta binario	Zebi	Zi	$(2^{10})^7$	1'180'591'620'717'411'303'424	$(10^3)^7$	18.06%
Yotta binario	Yobi	Yi	$(2^{10})^8$	1'208'925'819'614'629'174'706'176	$(10^3)^8$	20.89%

- Usando questi prefissi si può risolvere l'ambiguità
  - capacità di un disco fisso: 120 GB = 111.76 GiB,
  - capacità di un floppy: 1.406 MiB = 1.475 MB

# La codifica delle istruzioni

➤ Si segue lo schema presentato per i caratteri alfanumerici:

- **quali e quante** sono le istruzioni da codificare?
- qual è la **lunghezza** delle successioni di bit da utilizzare ?
- qual è la **corrispondenza** tra istruzioni e successioni di bit ?

Istruzioni aritmetico-logiche	
Codice	Istruzione
0111 1100	<b>ADD</b>
0111 1101	<b>SUB</b>
0111 1110	<b>AND</b>
... ..	... ..

Istruzioni per il trasferimento dati	
Codice	Istruzione
1110 1000	<b>LOAD</b>
1111 1000	<b>STORE</b>
... ..	... ..
... ..	... ..

Istruzioni di controllo	
Codice	Istruzione
0100 1001	<b>IF_EQ</b>
0100 1000	<b>GOTO</b>
0100 1100	<b>RETURN</b>
... ..	... ..

# Oltre al codice operativo

- ... è necessario far riferimento ai **dati** necessari per completare l'esecuzione dell'istruzione,
  - e.g. addizione: è necessario che sia specificato (anche **implicitamente**) dove leggere i due operandi da sommare e dove scrivere il risultato;
- il **numero** dei dati da specificare è variabile, in funzione delle istruzioni.

Codice Operativo	Destinazione	Sorgente 1	Sorgente 2	Estensione del codice operativo
------------------	--------------	------------	------------	---------------------------------

Codice Operativo	Destinazione	Sorgente 1	Operando (immediato)
------------------	--------------	------------	----------------------

Codice Operativo	Operando (immediato)
------------------	----------------------

# Rappresentazione dei numeri

# Rappresentazione dei numeri naturali

La rappresentazione dei numeri naturali  $(0, 1, 2, \dots)$  usata dai latini o dai greci (I, II, III, ...) era del tutto convenzionale, mentre quella da noi usata (trasmessa dagli arabi) in base dieci (o decimale) si basa su un **teorema fondamentale dell'aritmetica**.

Ciascun numero naturale  $(0, 1, 2, \dots)$  può essere rappresentato da una ed una sola successione finita di bit, detta **rappresentazione binaria** o in **base due**.

Questa possibilità è conseguenza di un importante **Teorema dell'Aritmetica**, lo stesso che consente l'usuale **rappresentazione decimale** dei numeri naturali, detta anche **rappresentazione in base 10**.

Questo Teorema è alla base anche della rappresentazioni dei naturali in base due (binaria) così come di altre basi (ottale, esadecimale, ecc.)

Per capire questo importante Teorema della rappresentazione dei numeri naturali, occorre distinguere tra:

i **numeri naturali**, i loro **nomi-concetti** e le loro **rappresentazioni**: queste ultime possono cambiare a seconda del supporto (base) scelto

## Teorema: Rappresentazione dei numeri naturali

Scelto un numero naturale  $N$  (detto *base*) maggiore o uguale a 2, ogni numero naturale  $M$  diverso da 0 può essere rappresentato in una sola maniera come somma finita di potenze decrescenti ( $k, k-1, \dots, 0$ ) di  $N$  moltiplicate per coefficienti ( $C_k, C_{k-1}, \dots, C_0$ ) minori di  $M$  ed  $N$ .

Ossia: scelta una base  $N \geq 2$ , per ogni numero naturale  $M \neq 0$ , esiste un solo  $k$  ed una sola  $k$ -pla di numeri (coefficienti)  $C_k, C_{k-1}, \dots, C_0$  minori di  $N$  e con  $C_k \neq 0$ , tale che:

$$M = (C_k \times N^k) + (C_{k-1} \times N^{k-1}) + \dots + (C_0 \times N^0)$$

**Esempi:**

se la base  $N = 10$ , ventitrè  $\Rightarrow (2 \times 10^1) + (3 \times 10^0) \Rightarrow 23$

se la base  $N = 2$ , ventitrè  $\Rightarrow (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) \Rightarrow 10111$

## Esempio 1: Rappresentazione dei numeri naturali

Assumiamo: base  $N=10$  ed  $M = \text{"duemilacentotto"}$

- La più grande potenza di 10 presente in  $N$  è la terza potenza di 10, cioè  $10^3$ ; questa compare ben 2 volte; otteniamo quindi il primo termine  $10^3 \times 2$  (= 2000).
- Togliendo  $(10^3 \times 2)$  da  $N$ , in quel che rimane (**centotto**), la seconda potenza di 10, ossia  $10^2$  compare 1 volta, quindi otteniamo il secondo termine  $10^2 \times 1$  (=100)
- Togliendo  $(10^3 \times 2)+(10^2 \times 1)$  da  $N$ , in quel che rimane (**otto**), la potenza prima di 10, ossia  $10^1$ , compare 0 volte, quindi otteniamo il terzo termine  $10^1 \times 0$  (=0)
- Togliendo infine  $(10^3 \times 2)+(10^2 \times 1)+(10^1 \times 0)$  da  $N$ , in quel che rimane (**otto**), la potenza nulla di 10, ossia  $10^0$  (=1) compare 8 volte, quindi otteniamo il quarto ed ultimo termine  $10^0 \times 8$  (=8)

Pertanto, **duemilacentotto in base 10** è rappresentato da:

$$(10^3 \times 2) + (10^2 \times 1) + (10^1 \times 0) + (10^0 \times 8) = 2108_{\text{dieci}}$$

## Esempio 2: Rappresentazione dei numeri naturali

Assumiamo: base  $N=2$  ed  $M = \text{"ventitrè"}$

- La più grande potenza di 2 presente in  $N$  è la quarta potenza di 2, cioè  $2^4$ ; questa compare 1 sola volta; otteniamo quindi il primo termine  $2^4 \times 1 (=16)$ .
- Togliendo  $(2^4 \times 1)$  da  $N$ , in quel che rimane (**sette**), la terza potenza di 2, ossia  $2^3$  compare 0 volte, quindi otteniamo il secondo termine  $2^3 \times 0 (=0)$
- Togliendo  $(2^4 \times 1) + (2^3 \times 0)$  da  $N$ , in quel che rimane (**sette**), la seconda potenza di 2, ossia  $2^2$ , compare 1 volta, quindi otteniamo il terzo termine  $2^2 \times 1 (=4)$
- Togliendo  $(2^4 \times 1) + (2^3 \times 0) + (2^2 \times 0)$  da  $N$ , in quel che rimane (**tre**), la prima potenza di 2, ossia  $2^1 (=2)$  compare 1 volta, quindi otteniamo il quarto termine  $2^1 \times 1 (=2)$
- Infine, togliendo  $(2^4 \times 1) + (2^3 \times 0) + (2^2 \times 0) + (2^1 \times 1)$  da  $N$ , in quel che rimane (**uno**), la potenza nulla di 2, ossia  $2^0 (=1)$  compare 1 volta, quindi otteniamo il quinto ed ultimo termine  $2^0 \times 1 (=1)$

Pertanto, **ventitrè in base 2** è rappresentato da:

$$(2^4 \times 1) + (2^3 \times 0) + (2^2 \times 1) + (2^1 \times 1) + (2^0 \times 1) = 10111_{\text{due}}$$

# Numeri naturali

Fissata la base, il Teorema assicura l'univocità della "lettura" (o "interpretazione" o "de-codifica") della sequenza di simboli (cifre) di cui si compone il numero dato

## ➤ Sistema di numerazione posizionale in base **b**

- $N = c_k c_{k-1} \dots c_0$  rappresenta  $c_k \times b^k + c_{k-1} \times b^{k-1} + \dots + c_0 \times b^0$ 
  - $b=10 \Rightarrow 1101_{\text{dieci}}$  indica  $1 \times 10^3 + 1 \times 10^2 + 0 \times 10 + 1 \times 10^0$
  - $b=2 \Rightarrow 1101_{\text{due}}$  indica  $1 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 \times 2^0$

## ➤ Conversione **binario** $\Rightarrow$ **decimale**

- basta scrivere il numero secondo la notazione posizionale utilizzando già il sistema decimale
- $b=2 \Rightarrow 1101_{\text{due}}$  indica  $1 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 \times 2^0 = 13_{\text{dieci}}$

# Conversione binario $\Rightarrow$ decimale

$$\begin{aligned}101100_{\text{due}} &= 1_{\text{dieci}} \times 2_{\text{dieci}}^5 + 0_{\text{dieci}} \times 2_{\text{dieci}}^4 + 1_{\text{dieci}} \times 2_{\text{dieci}}^3 + 1_{\text{dieci}} \times 2_{\text{dieci}}^2 + 0_{\text{dieci}} \times 2_{\text{dieci}}^1 + \\ &\quad + 0_{\text{dieci}} \times 2_{\text{dieci}}^0 = \\ &= 1_{\text{dieci}} \times 32_{\text{dieci}} + 0_{\text{dieci}} \times 16_{\text{dieci}} + 1_{\text{dieci}} \times 8_{\text{dieci}} + 1_{\text{dieci}} \times 4_{\text{dieci}} + 0_{\text{dieci}} \times 2_{\text{dieci}} \\ &\quad + 0_{\text{dieci}} \times 1_{\text{dieci}} = \\ &= 32_{\text{dieci}} + 8_{\text{dieci}} + 4_{\text{dieci}} = \\ &= 44_{\text{dieci}}\end{aligned}$$

$$\begin{aligned}101110101_{\text{due}} &= 1_{\text{dieci}} \times 2_{\text{dieci}}^8 + 0_{\text{dieci}} \times 2_{\text{dieci}}^7 + 1_{\text{dieci}} \times 2_{\text{dieci}}^6 + 1_{\text{dieci}} \times 2_{\text{dieci}}^5 + 1_{\text{dieci}} \times 2_{\text{dieci}}^4 + \\ &\quad 0_{\text{dieci}} \times 2_{\text{dieci}}^3 + 1_{\text{dieci}} \times 2_{\text{dieci}}^2 + 0_{\text{dieci}} \times 2_{\text{dieci}}^1 + 1_{\text{dieci}} \times 2_{\text{dieci}}^0 = \\ &= 1_{\text{dieci}} \times 256_{\text{dieci}} + 0_{\text{dieci}} \times 128_{\text{dieci}} + 1_{\text{dieci}} \times 64_{\text{dieci}} + 1_{\text{dieci}} \times 32_{\text{dieci}} + \\ &\quad 1_{\text{dieci}} \times 16_{\text{dieci}} + 0_{\text{dieci}} \times 8_{\text{dieci}} + 1_{\text{dieci}} \times 4_{\text{dieci}} + 0_{\text{dieci}} \times 2_{\text{dieci}} + 1_{\text{dieci}} \times 1_{\text{dieci}} = \\ &= 256_{\text{dieci}} + 64_{\text{dieci}} + 32_{\text{dieci}} + 16_{\text{dieci}} + 4_{\text{dieci}} + 1_{\text{dieci}} = \\ &= 373_{\text{dieci}}\end{aligned}$$

# Operazioni binarie fondamentali

- **Operazioni fondamentali: operazioni elementari sui bit.**
- **Sono definite le operazioni aritmetiche più le *operazioni logiche* (AND, OR, NOT).**
- **Le operazioni possono essere descritte in forma tabellare (esempi: +, x, AND, OR)**

		<b>+ (XOR) Disg. Excl.</b>	<b>OR Disg. Incl.</b>	<b>x</b>	<b>AND Congiunz.</b>
<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>
<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>0</b>
<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>

# Numeri binari: operazione +

- Operazioni di somma di numeri binari naturali.
- Con gli **8 bit** utilizzati negli esempi qui riportati si possono rappresentare  **$2^8$  numeri naturali**: da  **$0_{\text{dieci}}$**  a  **$255_{\text{dieci}}$** .
- Operazioni che producono un risultato maggiore provocano il superamento della capacità di rappresentazione (indicato in gergo dal termine inglese **overflow**).

$$\begin{array}{r} \phantom{00}11 \\ 00011001_{\text{due}} + \\ 00111100_{\text{due}} = \\ \hline 01010101_{\text{due}} \end{array} \qquad \begin{array}{r} 25_{\text{dieci}} + \\ 60_{\text{dieci}} = \\ \hline 85_{\text{dieci}} \end{array}$$

$$\begin{array}{r} \phantom{00}11 \\ 00101100_{\text{due}} + \\ 01001110_{\text{due}} = \\ \hline 01111010_{\text{due}} \end{array} \qquad \begin{array}{r} 44_{\text{dieci}} + \\ 78_{\text{dieci}} = \\ \hline 122_{\text{dieci}} \end{array}$$

# Numeri binari: operazioni x

- Operazioni di moltiplicazione dei numeri binari.
- Facilitate dal fatto che le moltiplicazioni per una cifra non richiedono calcoli ( $N \times 0 = 0$ ,  $N \times 1 = N$ )
- Le moltiplicazioni a più cifre si effettuano per somme successive (algoritmo manuale)

**Esempio:**       $0011_{\text{due}} \times 3_{\text{dieci}}$   
                   $0011_{\text{due}} = 3_{\text{dieci}}$

$$\begin{array}{r} \text{-----} \\ 0011 \\ 0011 \\ 0000 \\ 0000 \\ \text{-----} \\ 0001001_{\text{due}} \end{array} \quad 9_{\text{dieci}}$$

# Numeri Naturali: conversione **decimale** => **binario**

Si potrebbe utilizzare lo stesso metodo posizionale usato sopra

• B=10 =>  $345_{\text{dieci}} = 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$

• B=2 =>  $345_{\text{dieci}} = 011 \times 1010^{10} + 100 \times 1010^1 + 101 \times 1010^0$

```

1010x
1010=
-----
 0000
 1010
 0000
1010
-----
1100100x
  11=
-----
 1100100
1100100
-----
100101100
    
```

```

1010x
 100=
-----
 0000
 0000
 1010
-----
101000
    
```

101

Quindi:  $345_{\text{dieci}} = 101011001_{\text{due}} = (100101100 + 101000 + 101)$

## Numeri Naturali: conversione **decimale => binario**

### **Esempio: da decimale a binario**

$$\begin{aligned} 565_{\text{dieci}} &= 5_{\text{dieci}} \times 10_{\text{dieci}}^2 + 6_{\text{dieci}} \times 10_{\text{dieci}}^1 + 5_{\text{dieci}} \times 10_{\text{dieci}}^0 = \\ &= 101_{\text{due}} \times 1010_{\text{due}}^2 + 110_{\text{due}} \times 1010_{\text{due}}^1 + 101_{\text{dieci}} \times 1010_{\text{due}}^0 = \\ &= 101_{\text{due}} \times 1100100_{\text{due}} + 110_{\text{due}} \times 1010_{\text{due}} + 101_{\text{due}} \times 1_{\text{due}} = \\ &= 111110100_{\text{due}} + 111100_{\text{due}} + 101_{\text{due}} = \\ &= 1000110101_{\text{due}} \end{aligned}$$

**...ma è molto complesso (dobbiamo pensare in binario!)**

# Conversione decimale $\Rightarrow$ binario

- Sistema di numerazione posizionale in base B che, in questo contesto si può ipotizzare diversa da dieci. Ecco un generico numero di n cifre in base B

$$C_{n-1}C_{n-2}\dots C_1C_0 = C_{n-1}\times B^{n-1} + C_{n-2}\times B^{n-2} + \dots + C_1\times B^1 + C_0\times B^0$$

$$C_{n-1}C_{n-2}\dots C_1C_0 = C_{n-1}\times B^{n-1} + C_{n-2}\times B^{n-2} + \dots + C_1\times B + C_0 \quad (\text{infatti } B^1 = B \text{ e } B^0 = 1)$$

- Dividendo il numero per il valore della base, si ottiene :

$$(C_{n-1}\times B^{n-1} + C_{n-2}\times B^{n-2} + \dots + C_1\times B + C_0)/B = C_{n-1}\times B^{n-2} + C_{n-2}\times B^{n-3} + \dots + C_1 + C_0/B$$

- che può essere scomposto in modo da evidenziare quoziente e resto:

$$\text{quoziente} = C_{n-1}\times B^{n-2} + C_{n-2}\times B^{n-3} + \dots + C_1 \quad \text{resto} = C_0$$

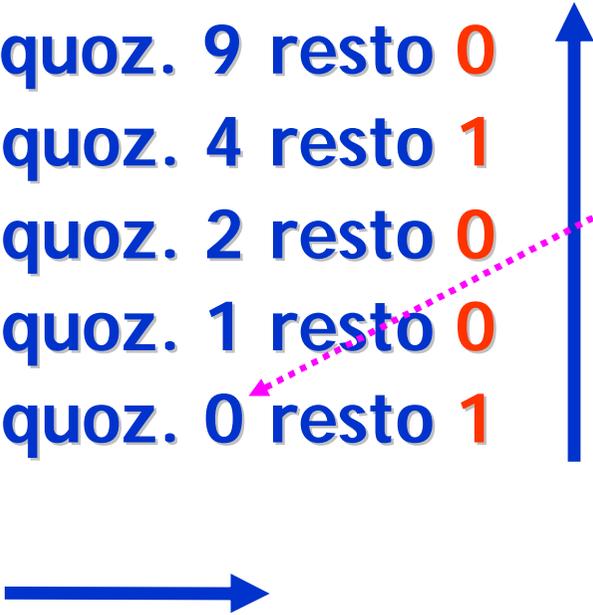
- **Il resto della divisione corrisponde all'ultima cifra della rappresentazione in base B del numero**, ma il suo valore è indipendente dalla base B che si utilizza per effettuare i conti.
- Applicando lo stesso procedimento al quoziente si ottiene la **penultima** cifra della rappresentazione in base B
- Iterando la procedura (fino al quoziente 0) otteniamo le altre cifre.

# Conversione decimale $\Rightarrow$ binario

(metodo "semplice")

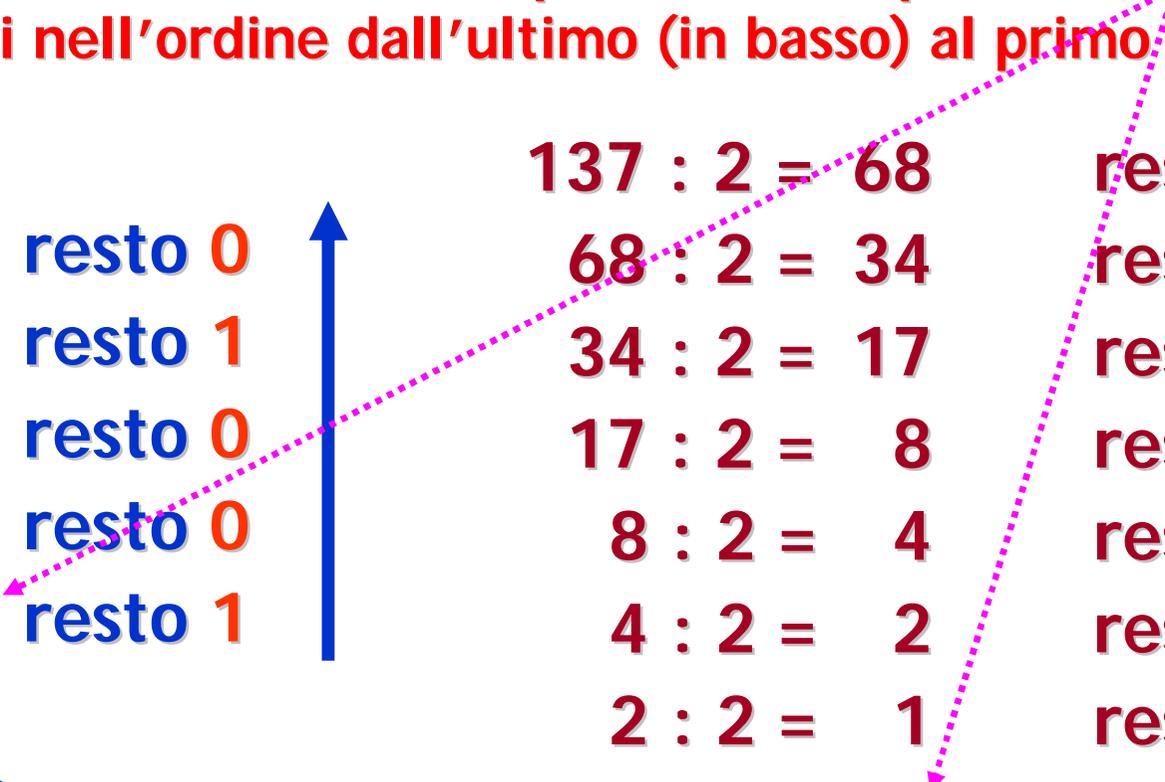
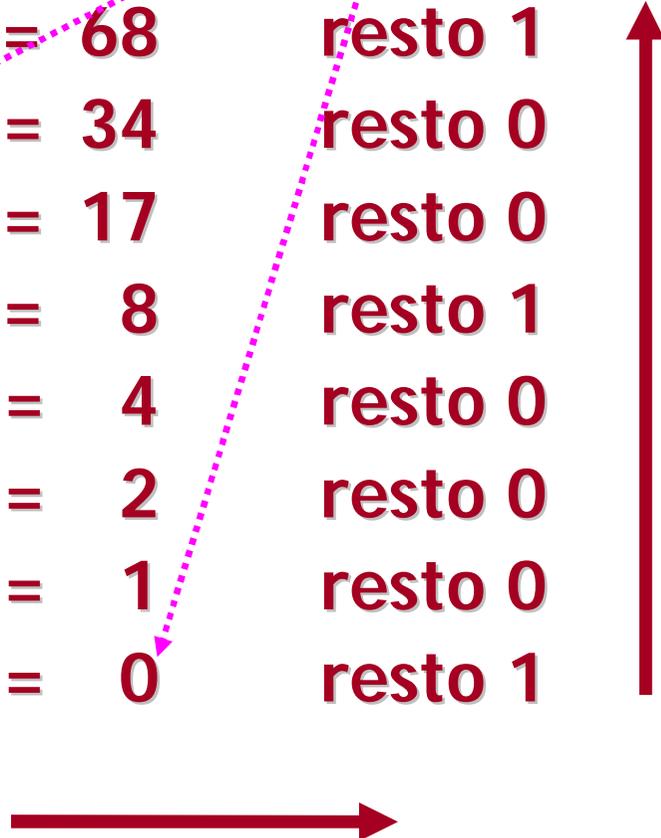
Si calcolano i resti delle divisioni per 2 fino al quoziente 0  
Si considerano i resti nell'ordine dall'ultimo (in basso) al primo (in alto)

$18 : 2 =$	quoz. 9	resto 0
$9 : 2 =$	quoz. 4	resto 1
$4 : 2 =$	quoz. 2	resto 0
$2 : 2 =$	quoz. 1	resto 0
$1 : 2 =$	quoz. 0	resto 1



10010

$137 : 2 =$	68	resto 1
$68 : 2 =$	34	resto 0
$34 : 2 =$	17	resto 0
$17 : 2 =$	8	resto 1
$8 : 2 =$	4	resto 0
$4 : 2 =$	2	resto 0
$2 : 2 =$	1	resto 0
$1 : 2 =$	0	resto 1



10001001

# Conversione decimale $\Rightarrow$ binario

					(cifra binaria meno significativa)
$573_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$286_{\text{dieci}}$	<i>resto</i> $1_{\text{dieci}}$
$286_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$143_{\text{dieci}}$	<i>resto</i> $0_{\text{dieci}}$
$143_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$71_{\text{dieci}}$	<i>resto</i> $1_{\text{dieci}}$
$71_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$35_{\text{dieci}}$	<i>resto</i> $1_{\text{dieci}}$
$35_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$17_{\text{dieci}}$	<i>resto</i> $1_{\text{dieci}}$
$17_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$8_{\text{dieci}}$	<i>resto</i> $1_{\text{dieci}}$
$8_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$4_{\text{dieci}}$	<i>resto</i> $0_{\text{dieci}}$
$4_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$2_{\text{dieci}}$	<i>resto</i> $0_{\text{dieci}}$
$2_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$1_{\text{dieci}}$	<i>resto</i> $0_{\text{dieci}}$
$1_{\text{dieci}}$	$: 2_{\text{dieci}}$	$\Rightarrow$	<i>quoziente</i>	$0_{\text{dieci}}$	<i>resto</i> $1_{\text{dieci}}$
					(cifra binaria più significativa)



$$573_{\text{dieci}} = 1\ 000\ 111\ 101_{\text{due}}$$

# Codifica ed ambiguità

- $10_{\text{due}} \Rightarrow 2_{\text{dieci}}$
- $010_{\text{due}} \Rightarrow 2_{\text{dieci}}$
- **Problema:** abbiamo due configurazioni binarie (due codici binari) che corrispondono (o rappresentano) lo stesso numero naturale; dunque, la codifica binaria è ambiguità?
- **Soluzione:** no, se facciamo l'ipotesi di fissare la lunghezza (numero di bit) dei codici binari che rappresentano o corrispondono ai numeri naturali
- **Domanda:** con N bit, quanti numeri naturali posso rappresentare?
- **Risposta:** da 0 a  $2^n - 1$

# Numeri naturali binari nei calcolatori

- Per la codifica dei numeri **naturali** (interi positivi) si utilizzano abitualmente successioni di **32 bit** (4 byte) con cui si possono rappresentare i numeri compresi tra **0** e  **$2^{32}-1 = 4'294'967'295 \approx 4 \times 10^9$** .
- Raddoppiando la lunghezza delle successioni il massimo numero rappresentabile cresce **esponenzialmente**: passando da 32 a 64 bit il massimo numero rappresentabile diventa  **$2^{64}-1 \approx 16 \times 10^{18} = 1.6 \times 10^{19}$**  (cresce di 10 ordini di grandezza).
- A causa del minor numero di simboli dell'alfabeto binario rispetto a quello decimale, **un numero codificato in notazione binaria richiede più cifre rispetto a quelle impiegate in notazione decimale.**

# Ottali

➤ Per rappresentare sinteticamente i valori binari

➤ **Ottali (base b = 8)** Alfabeto: cifre comprese tra 0 e 7

➤ Esempio da Decimale a Ottale:

<b>236</b> :	8	quoz. 29	resto <b>4</b>
29:	8	quoz. 3	resto <b>5</b>
3:	8	quoz. 0	resto <b>3</b>



Lettura (posizionale) Ottale:

$$354_{\text{otto}} \Rightarrow 3 \times 10^2 + 5 \times 10^1 + 4 \times 10^0 = 300 + 50 + 4 = 354_{\text{otto}}$$

➤ Trasformazione da Ottale a Decimale:

$$354_{\text{otto}} \Rightarrow 3 \times 8^2 + 5 \times 8^1 + 4 \times 8^0 = 192 + 40 + 4 = 236_{\text{dieci}}$$

$$1461_{\text{otto}} \Rightarrow 1 \times 8^3 + 4 \times 8^2 + 6 \times 8^1 + 1 \times 8^0 = 512 + 256 + 48 + 1 = 817_{\text{dieci}}$$

Ogni cifra ottale corrisponde a tre cifre binarie  $8=2^3$ :

$$236_{\text{dieci}} = 11101100_{\text{due}} = [011] [101] [100] = 354_{\text{otto}}$$

$$453_{\text{dieci}} = 1100110001_{\text{due}} = [001] [100] [110] [001] = 1461_{\text{otto}}$$

# Esadecimale

➤ Per rappresentare sinteticamente i valori binari

➤ **Esadecimale (base  $b = 16$ )** : cifre 0 - 9 + lettere A - F

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 (decimale)

0 1 2 3 4 5 6 7 8 9 A B C D E F (alfabeto ottale)

➤ **Da Decimale a Esadecimale:**  $236 : 16$  quoz. 14 resto 12 = C  
 $14 : 16$  quoz. 0 resto 14 = E

**Da Esadecimale a Decimale:**

$$EC_{\text{sedici}} = 14 \times 16^1 + 12 \times 16^0 = 224 + 12 = 236_{\text{dieci}}$$

$$331_{\text{sedici}} = 3 \times 16^2 + 3 \times 16^1 + 1 \times 16^0 = 768 + 48 + 1 = 817_{\text{dieci}}$$

➤ **Ogni cifra esadecimale corrisponde a quattro cifre binarie  $16=2^4$ :**

$$11101100_{\text{due}} = [1110] [1100] = EC_{\text{sedici}}$$

$$1100110001_{\text{due}} = [0011] [0011] [0001] = 331_{\text{sedici}}$$

# Diverse codifiche/interpretazioni

Codice	Nat	MS
0000	0	<b>+0</b>
0001	1	+1
0010	2	+2
0011	3	+3
0100	4	+4
0101	5	+5
0110	6	+6
0111	7	+7

Codice	Nat	MS
1000	8	<b>-0</b>
1001	9	-1
1010	10	-2
1011	11	-3
1100	12	-4
1101	13	-5
1110	14	-6
1111	15	-7

# **Analogico vs digitale**

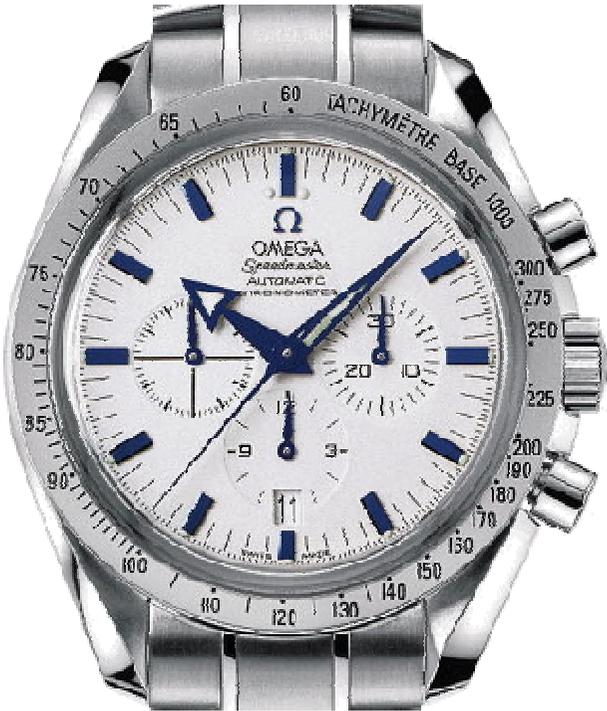
# Informazione “classificatoria” e “più che classificatoria”

- **Informazione classificatoria:**  
“è questo, ma avrebbe potuto essere quest’altro”.
- **Informazione più che classificatoria:**
  - riconoscere **distinzioni**;
  - stabilire una **relazione d’ordine** (“questo è maggiore di quest’altro”);
  - stabilire una **metrica** (“questo è distante un certo valore da quest’altro”).
- **L’insieme delle entità di informazione ha una struttura.**
  - La **struttura** dice “cosa si può fare” con le entità di informazione dell’insieme, in termini di operazioni di combinazione e di confronto.
  - Si tratta di informazione su informazione (**meta-informazione**) in presenza della quale l’insieme delle entità di informazione diventa un **sistema**, cioè appunto un “**insieme con struttura**”.

# Due alternative

- **Meta-informazione *esplicita* nel supporto:**
  - il supporto ha una struttura corrispondente (*analoga*) a quella presente tra entità di informazione.
- **Meta-informazione *implicita* nella regola di codifica:**
  - al supporto si richiede solo di avere configurazioni molteplici e distinguibili l'una dall'altra,
  - la meta-informazione è definita in modo estensionale nella regola di codifica.
- **Cosa succede nei due casi se si aggiungono i "mezzi punti"?**

# Analogico e digitale



➤ **Meta-informazione  
esplicita nel supporto**

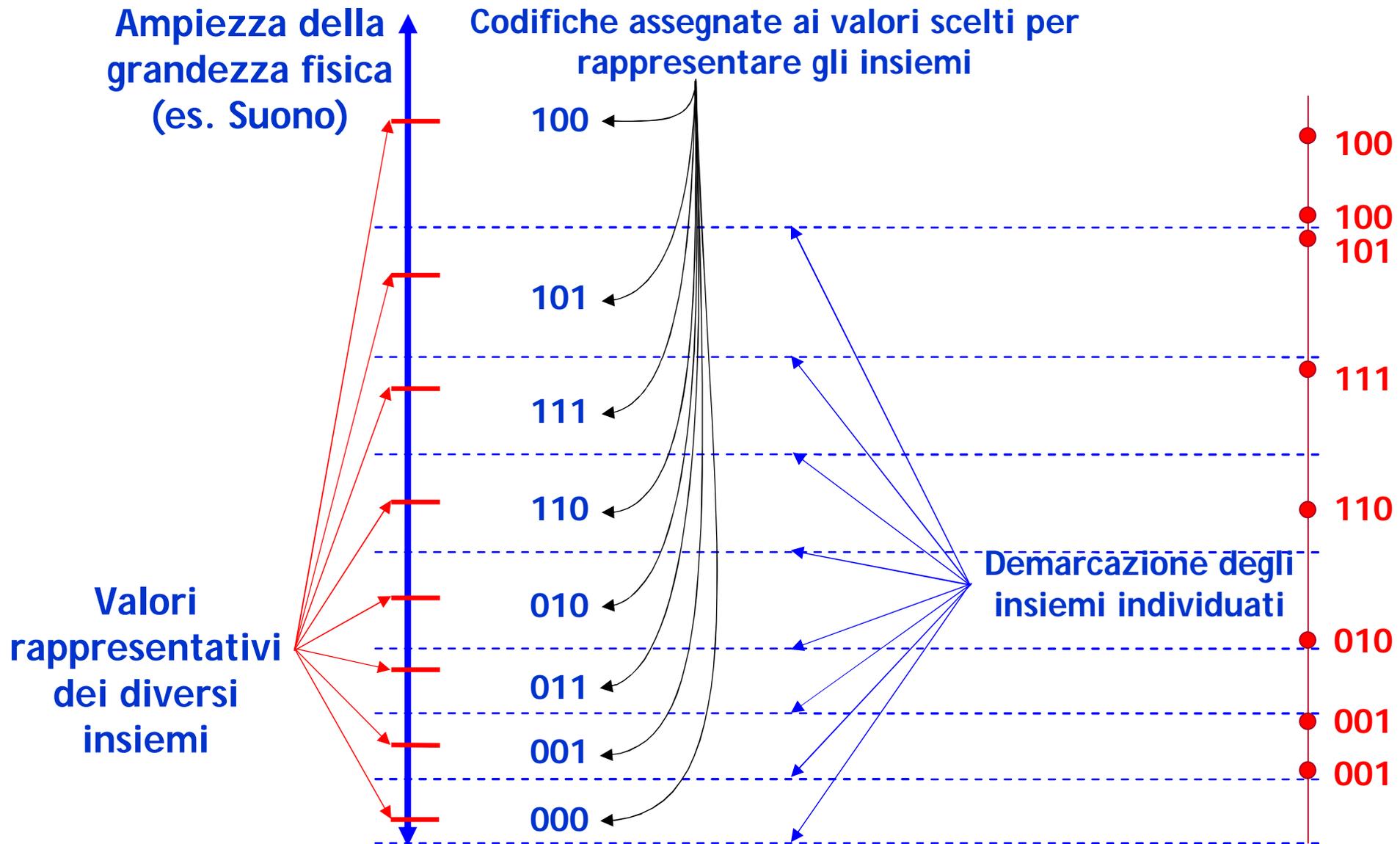
➤ **Codifica ANALOGICA**



➤ **Meta-informazione  
implicita nella codifica**

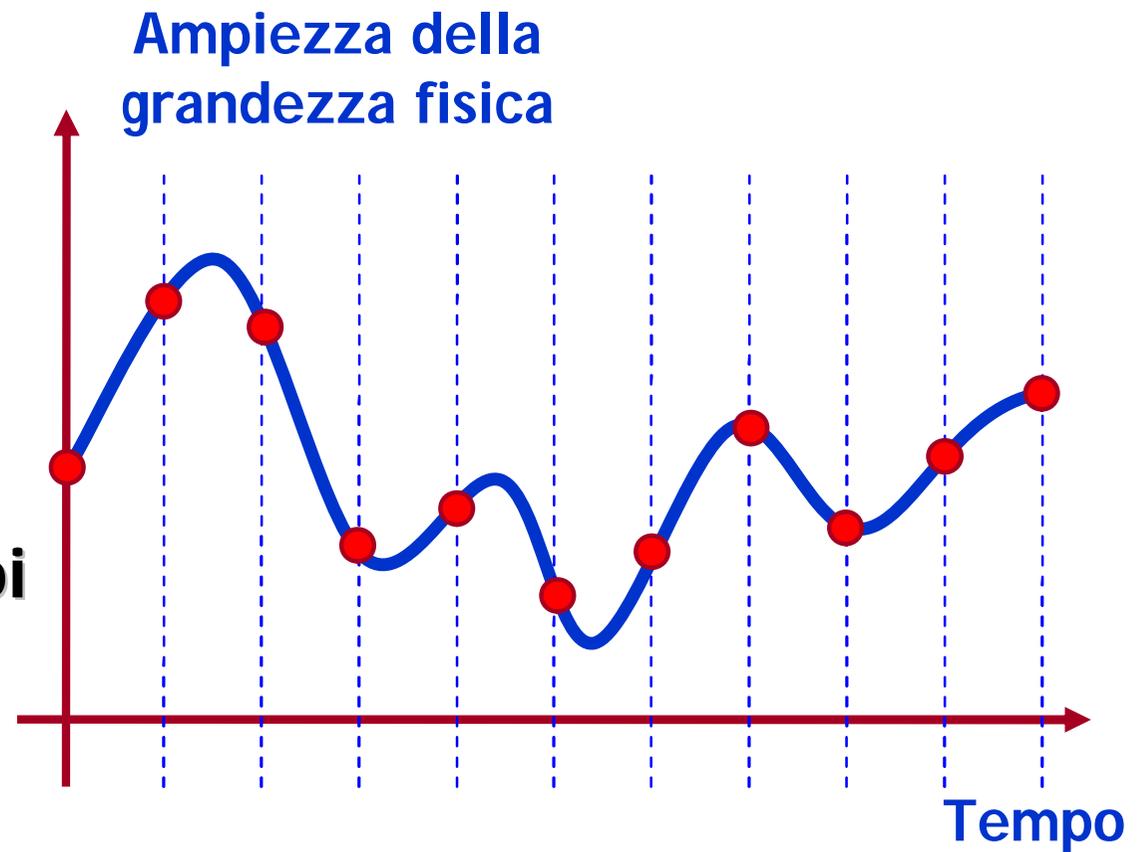
➤ **Codifica DIGITALE**

# Da analogico a digitale: 1. la quantizzazione



# Da analogico a digitale: 2. il campionamento

- La grandezza varia nel tempo e non può essere rappresentata da un solo valore.
- I valori di riferimento debbono essere rilevati in diversi istanti di tempo (**frequenza di campionamento**).
- La quantizzazione deve poi essere ripetuta per ogni valore campionato.



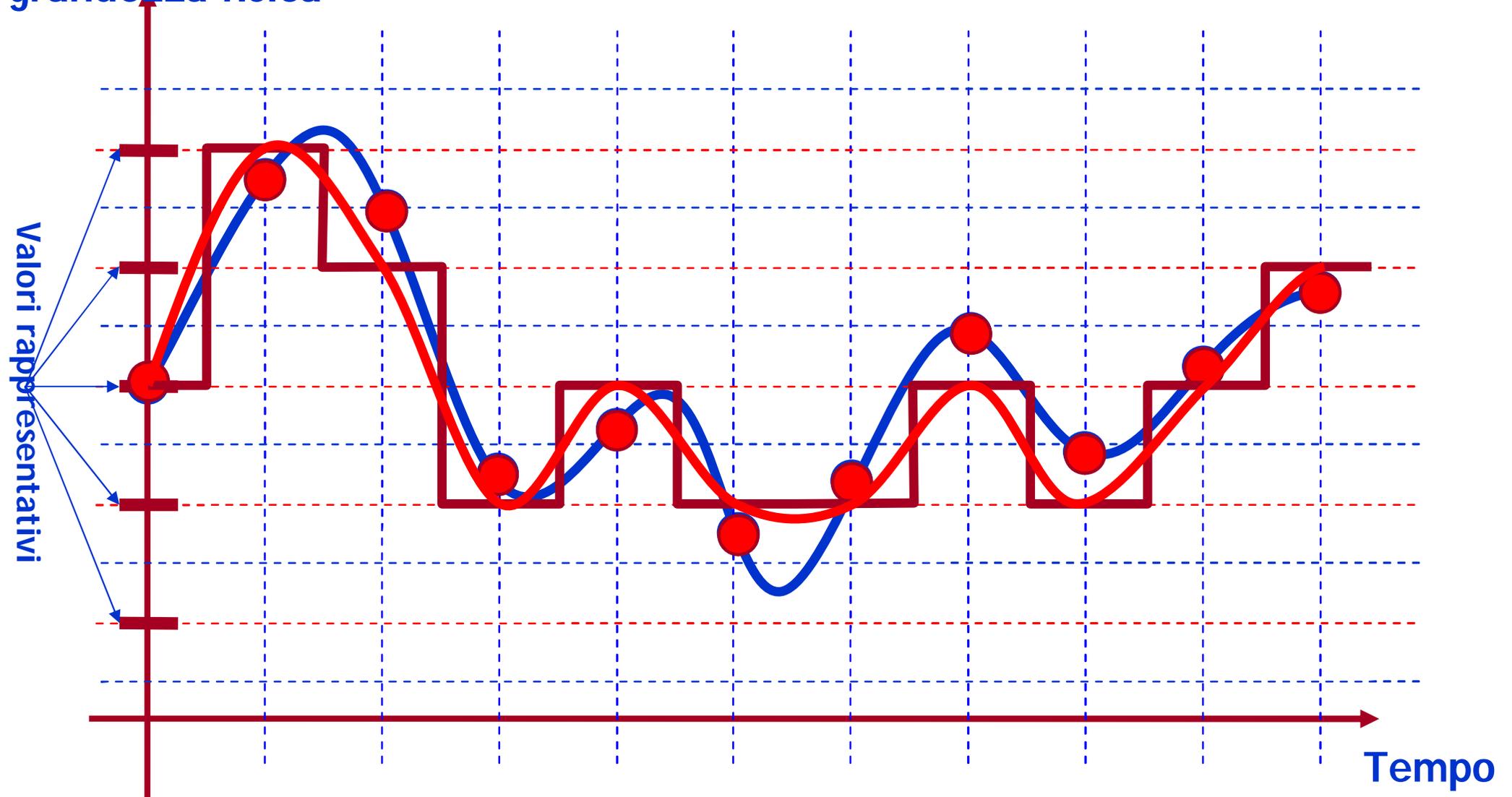
# Da analogico a digitale: quantizzazione e campionamento

- il **parametro** che caratterizza la **quantizzazione** (suddivisione di una grandezza fisica in intervalli) è il **numero di BIT**
- il **parametro** che caratterizza il **campionamento** è la **frequenza di campionamento**, cioè il **numero di campioni** che vengono acquisiti nell'unità di tempo e si misura in Hz (Hertz) al secondo.

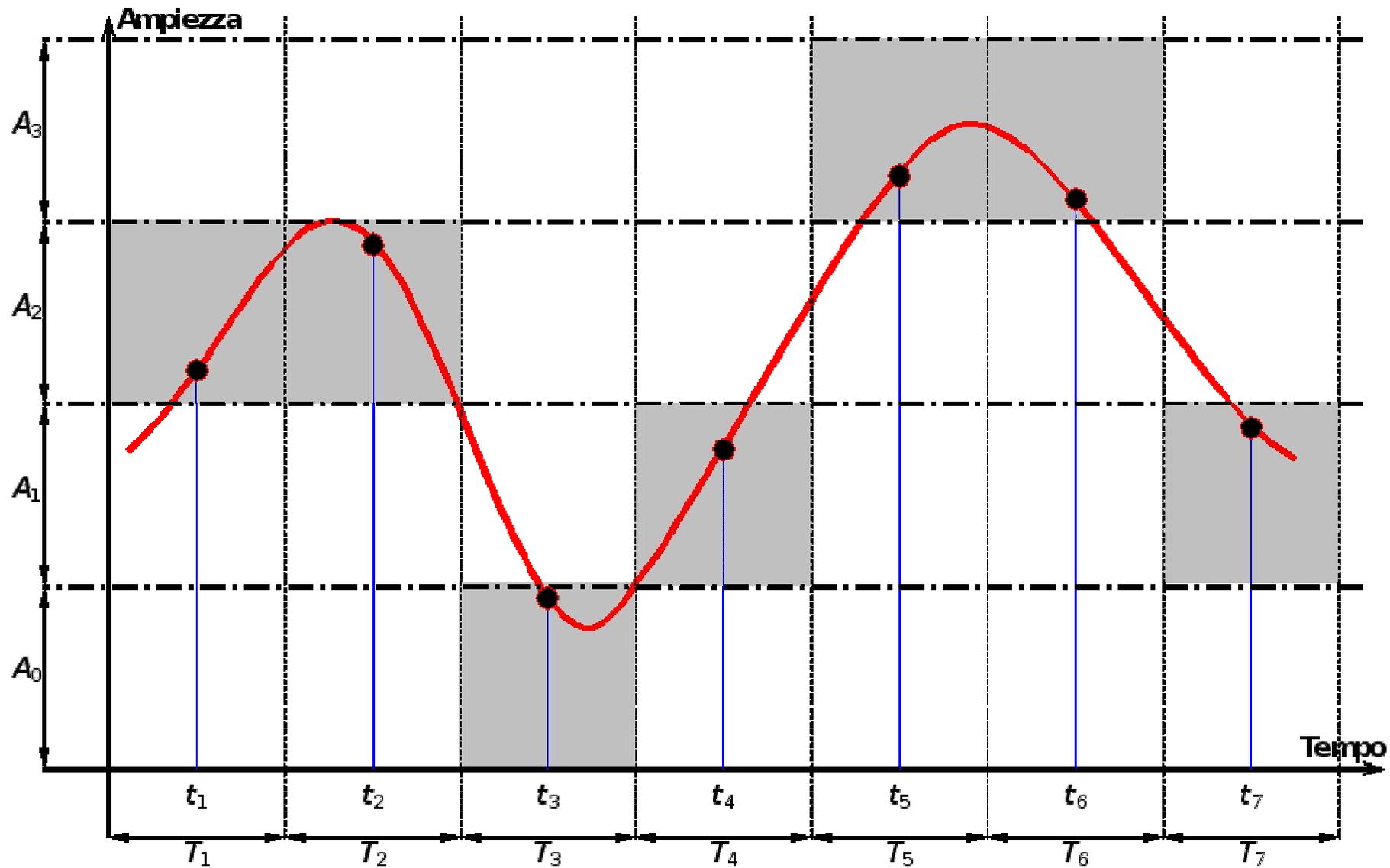
**Esempio:** un campionamento effettuato con frequenza di **5 Hz**, consiste nell'acquisizione di **5 campioni al secondo**

# Campionamento e quantizzazione

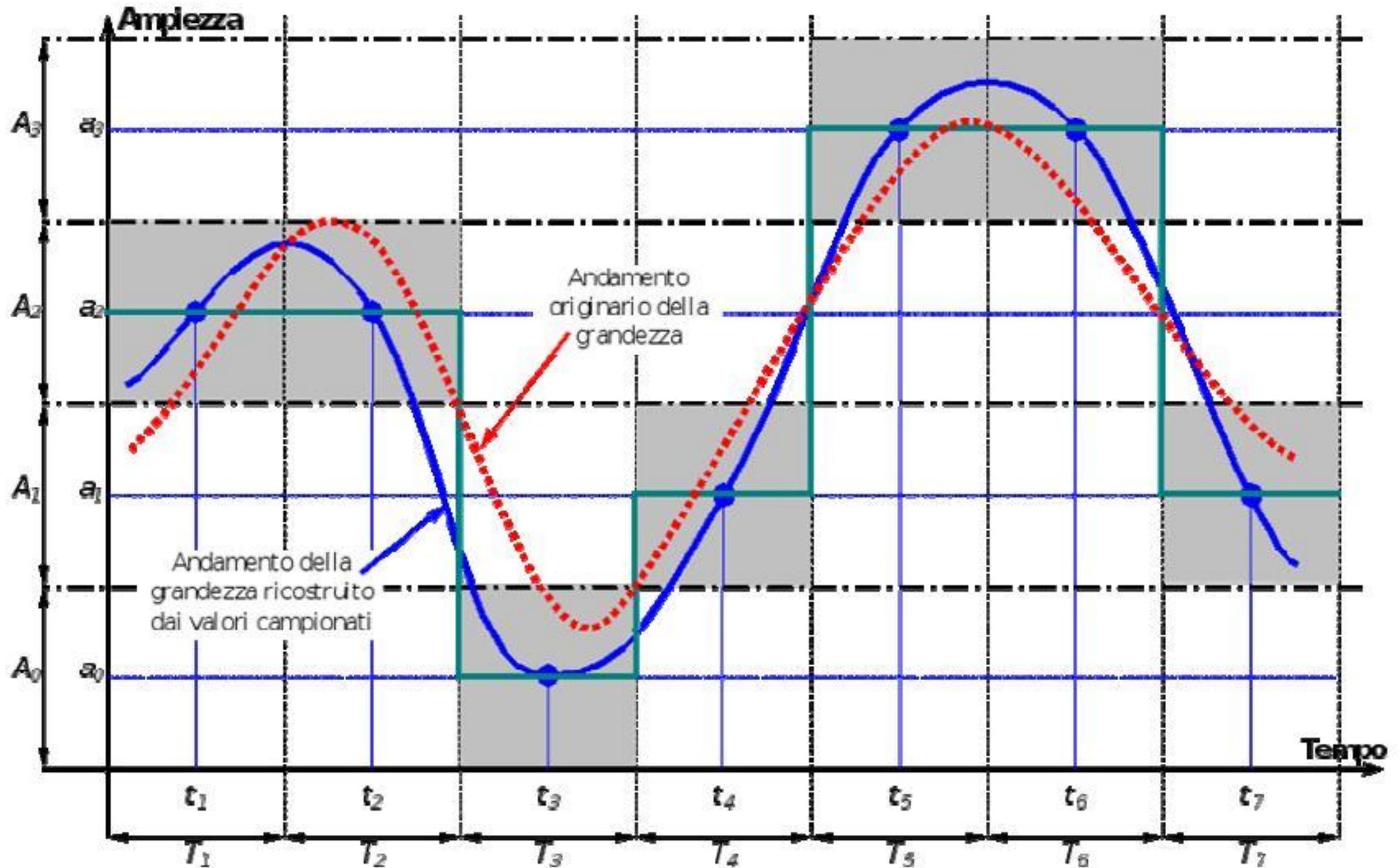
Ampiezza della  
grandezza fisica



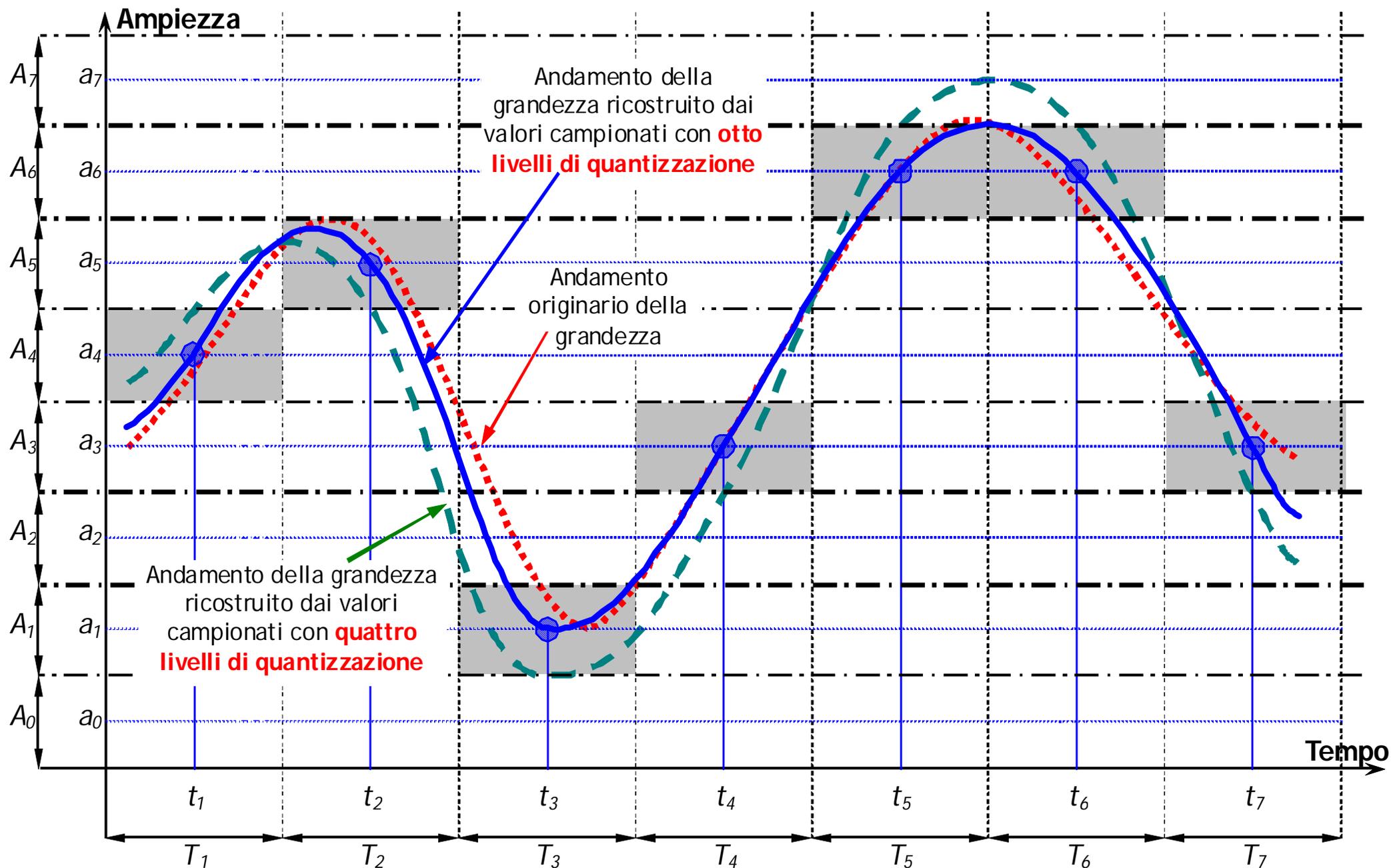
# Campionamento a 2 BIT (4 configurazioni) e Quantizzazione a 7 Hertz (7 campioni al secondo)



# Ricostruzione



# Quantizzazione su più livelli (8 BIT)



# Suono digitale

- Formato standard per i CD audio
  - frequenza di campionamento di 44'100 Hz (c.ca 44 KHz)
  - quantizzazione su 65'536 livelli (16 bit)  
(un campione viene codificato su 16 bit)
- Un secondo di musica stereo richiede 44'100 campioni di 16 bit (2 byte) ciascuno per due canali, quindi 176'400 byte (cioè,  $44.100 \times 2 \text{ bytes} \times 2 \text{ canali}$ ).
- L'errore che si commette nella ricostruzione del segnale sonoro è difficilmente rilevabile da parte di un orecchio umano.

# Foto digitali

- Per la codifica digitale delle immagini le operazioni di campionamento e quantizzazione si applicano nello **spazio** invece che nel **tempo**.
- Il **campionamento** consiste nel dividere l'immagine in sottoinsiemi (*pixel*, cioè *picture element*), per ognuno dei quali si dovrà prelevare un campione che si considera rappresentativo del colore di tutto il sottoinsieme.
- La **quantizzazione** è la codifica del colore associato a ogni pixel: i più recenti formati utilizzano **32 bit** (4 byte) per pixel: 8 bit per ognuna delle tre componenti fondamentali (**RGB: red, green, blue**) e altri 8 per gestire le **trasparenze**.
- **Memoria necessaria per immagini non compresse (bitmap)**
  - per un'immagine di 640×480 pixel servono 1'228'800 byte;
  - per un'immagine di 800×600 pixel servono 1'920'000 byte;
  - per un'immagine di 1024×768 pixel servono 3'145'728 byte;
  - per un'immagine di 1280×1024 pixel servono 5'242'880 byte;
  - per un'immagine di 1600×1200 pixel servono 7'680'000 byte;
  - ...

# Il successo del digitale

- **Rumore:** effetto dell'ambiente sul supporto.
- **Quanto un supporto è "immune" al rumore?**
  - **Codifica analogica:** ogni configurazione è lecita dal punto di vista informativo e quindi risulta impossibile distinguere il rumore dal segnale.
  - **Codifica digitale:** un valore binario è associato a un insieme di **configurazioni valide** quindi si può
    - riconoscere il rumore che porta in configurazioni non lecite
    - trascurare il rumore che non fa uscire il segnale dall'insieme associato alla stessa configurazione



# Come ridurre il numero di bit

## ➤ Esempio

- **successione di un milione di caratteri**, ognuno scelto dall'insieme **{A, C, G, T}**;
- la frequenza dei quattro caratteri all'interno della successione non è uguale: **A** si presenta nel **50%** dei casi, **C** nel **25%**, **G** e **T** solo nel **12.5%** dei casi.

## ➤ Codifica digitale a **lunghezza costante**

- due bit per ciascuno dei simboli, per esempio: **A = 00**, **C = 01**, **G = 10** e **T = 11**;
- la lunghezza complessiva della successione è quindi pari a **2 milioni di bit**.

## ➤ Codifica a **lunghezza variabile** (che tenga conto della distribuzione)

- **A = 0**, **C = 10**, **G = 110** e **T = 111**;
- la lunghezza complessiva della successione è di **1.75 milioni di bit**  
( $1 \times 50\% + 2 \times 25\% + 3 \times 12.5\% + 3 \times 12.5\%$ ) bit/carattere  $\times$  1 milione di caratteri

## ➤ Il cambiamento di codifica permette di ridurre il numero di bit utilizzato senza perdere informazione.

# La compressione dei dati

- **Gli algoritmi di compressione dei dati possono essere distinti in due categorie fondamentali**
  - Compressione **lossless**, se **non provoca** la perdita di informazione
  - Compressione **lossy**, se **provoca** la perdita di informazione
- **Gli algoritmi di compressione lossless sfruttano le regolarità nei dati**
  - **RLE (Run Length Encoding)**: successioni di **n simboli uguali** vengono rappresentati con una coppia **<simbolo, n>** (esempio: Fax)
  - **Tecniche basate su dizionario**: sequenze di simboli ripetute trovate analizzando i dati sono sostituite con simboli elementari memorizzando le corrispondenze in un dizionario
- **Gli algoritmi di compressione lossy sono specifici per i diversi domini applicativi.**

# Un semplice esempio con dizionario

## ➤ **Compressione lossless** con tecnica basata su un dizionario

- **Testo di esempio:**

*"I re di Francia della dinastia Carolingia sono: Carlo II, Luigi II di Francia, Luigi III di Francia, Carlomanno di Francia, Carlo III detto il grosso, Odo, Carlo III detto il semplice, Roberto I di Francia, Rodolfo Duca di Borgogna, Luigi IV di Francia, Lotario di Francia, Luigi V di Francia"* (lunghezza: 292 caratteri)

## ➤ **Analisi delle regolarità presenti nel testo:**

- **Si individuano le sequenze (parole) ripetute contando le ripetizioni e si compila il dizionario (vedere tabella)**
- **Si assegna un simbolo che la sostituisce ad ogni parola**
- Il testo diventa:  
*"I re 1 2 della 1nastia Carolingia sono: 5 3, 4 3 1 2, 4 3I 1 2, 5manno 1 2, 5 3I 6 7 grosso, Odo, 5 3I 6 7 semplice, Roberto I 1 2, Rodolfo Duca 1 Borgogna, 4 IV 1 2, Lotario 1 2, 4 V 1 2"* (lunghezza: 187 caratteri + 35 caratteri per il dizionario = 222 caratteri - 76% della lunghezza originaria)
- Un testo più lungo permette una efficienza maggiore!

Indice	Parola	N
1	di	10
2	Francia	8
3	II	5
4	Luigi	4
5	Carlo	4
6	detto	2
7	il	2