L’informazione e la sua codifica

Corso di

Architetture dell’Informazione e della Comunicazione

Roberto Maieli – Roma Tre

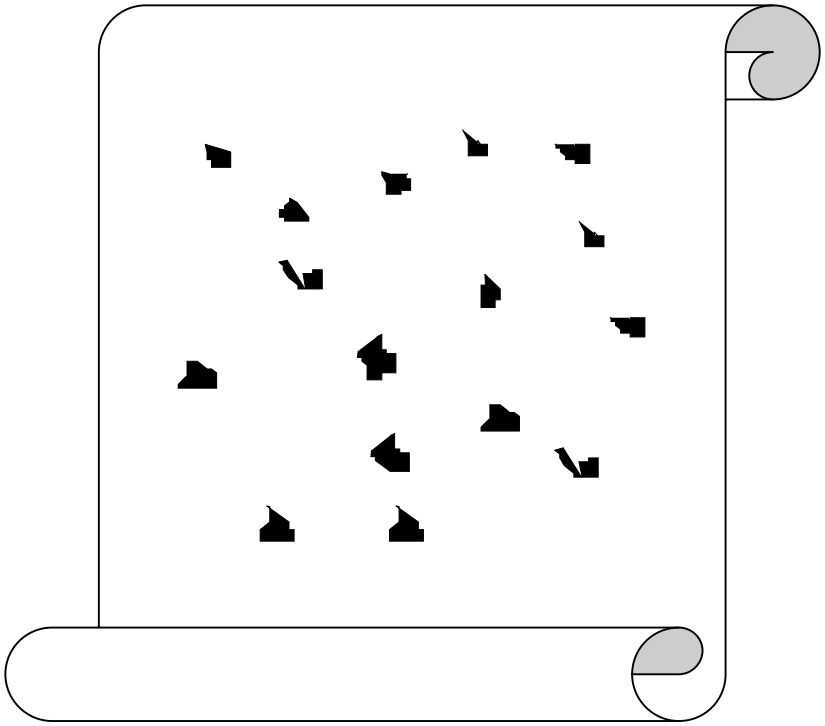
Informazione e Informatica

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

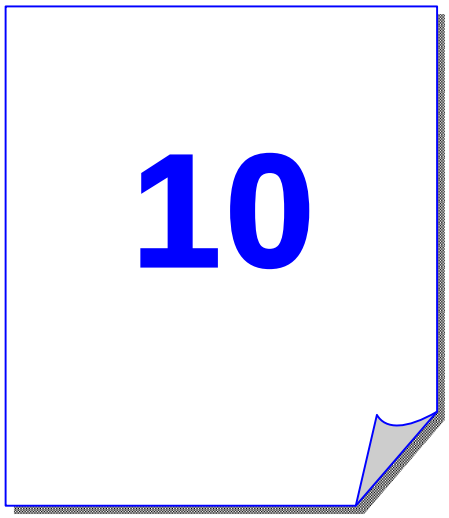
Informazione e supporto

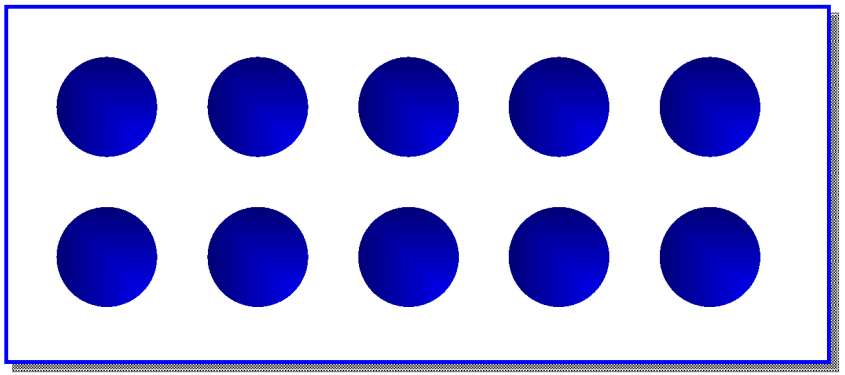
- L'informazione è "portata da", o "trasmessa su", o "memorizzata in", o "contenuta in" qualcosa; questo "qualcosa" però non è l'informazione stessa.

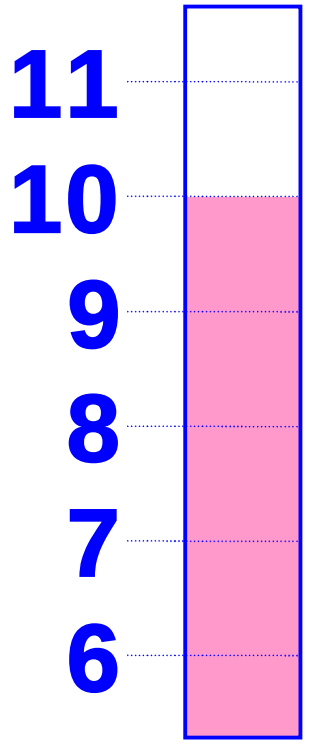
- Ogni supporto ha le sue caratteristiche in quanto mezzo su cui può essere scritta dell'informazione.
- Esempi: l'aria, un CD sono supporti

Informazione e supporti

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







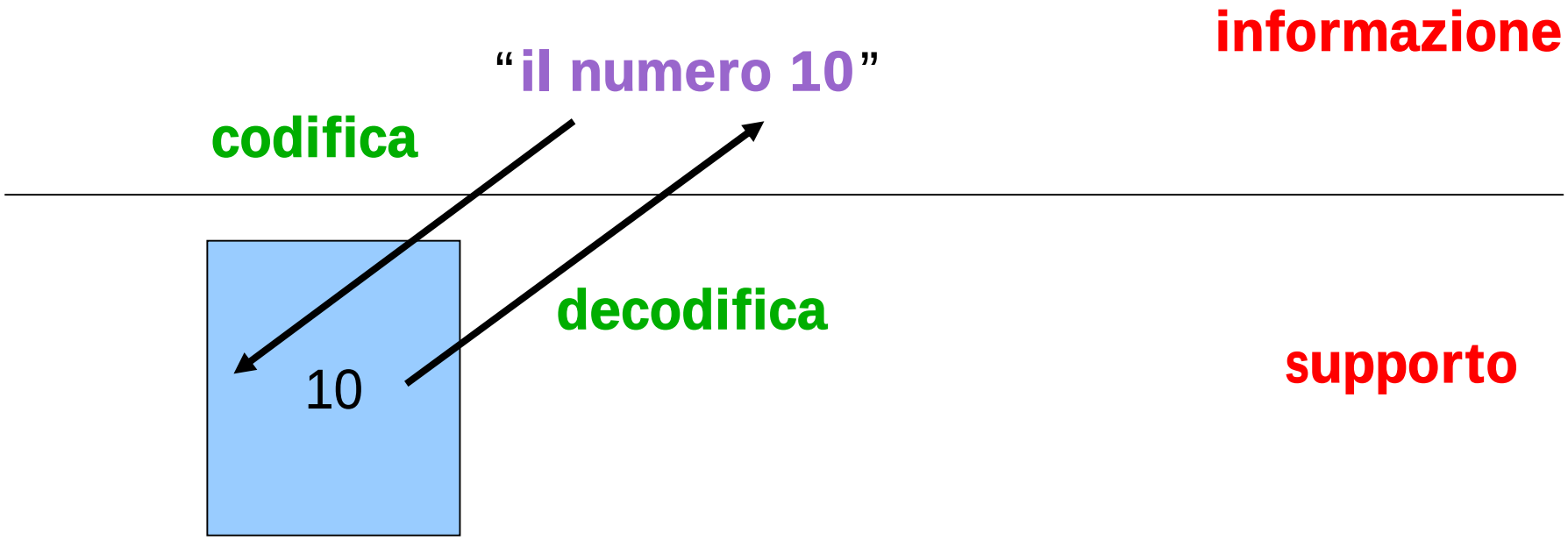
Denotano la stessa informazione ma significano forse **cose diverse**

Informazione e supporti

Lo stesso supporto può portare informazioni differenti.

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: 1ª condizione

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

- **Cosa rappresenta ciascuna configurazione?**

Configurazioni e codici

- **A ogni configurazione del supporto deve essere associata un'entità di informazione:**
  - Prima Configurazione = interruttore "**ON**" = "**Divina Commedia**";
  - Seconda Configurazione = interruttore "**OFF**" = "**I Promessi Sposi**".

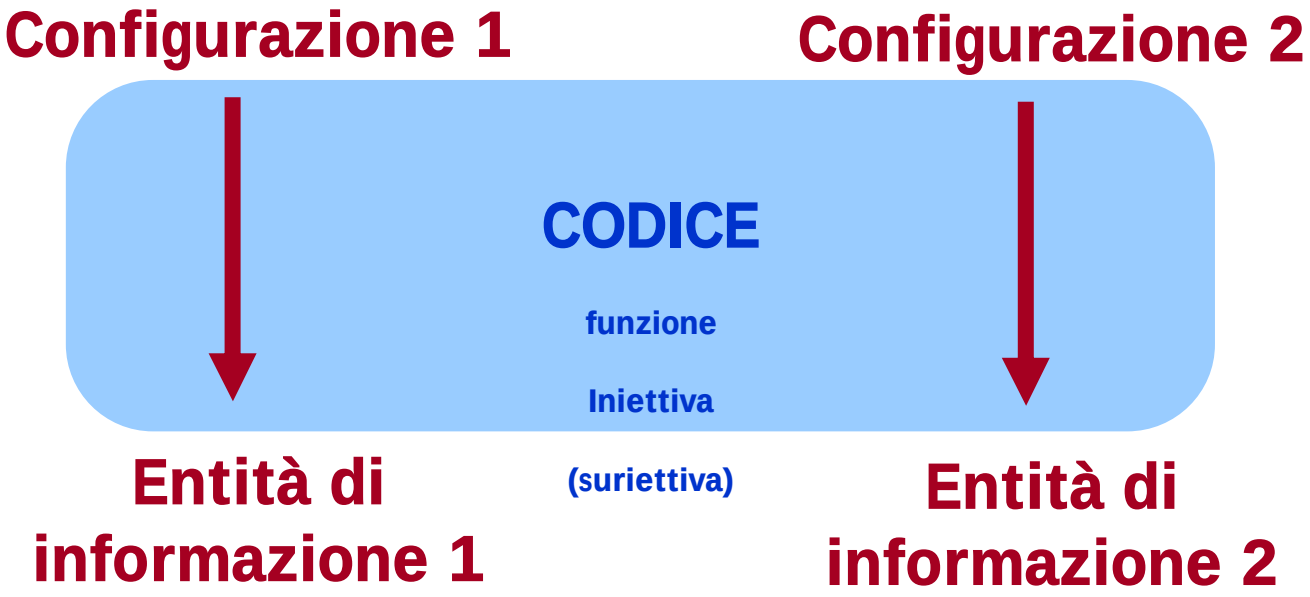
- **Per interpretare le differenti configurazioni del supporto in termini di informazione è necessario conoscere il codice (cioè la regola) che a ogni configurazione ammessa del supporto associa un'entità di informazione.**

- **La definizione di un codice comporta che siano identificati in modo non ambiguo l'insieme delle possibili configurazioni del supporto e l'insieme delle possibili entità di informazione a cui ci si vuole riferire.**

- **Variando il codice è possibile riferirsi a entità di informazione differenti utilizzando uno stesso supporto fisico.**

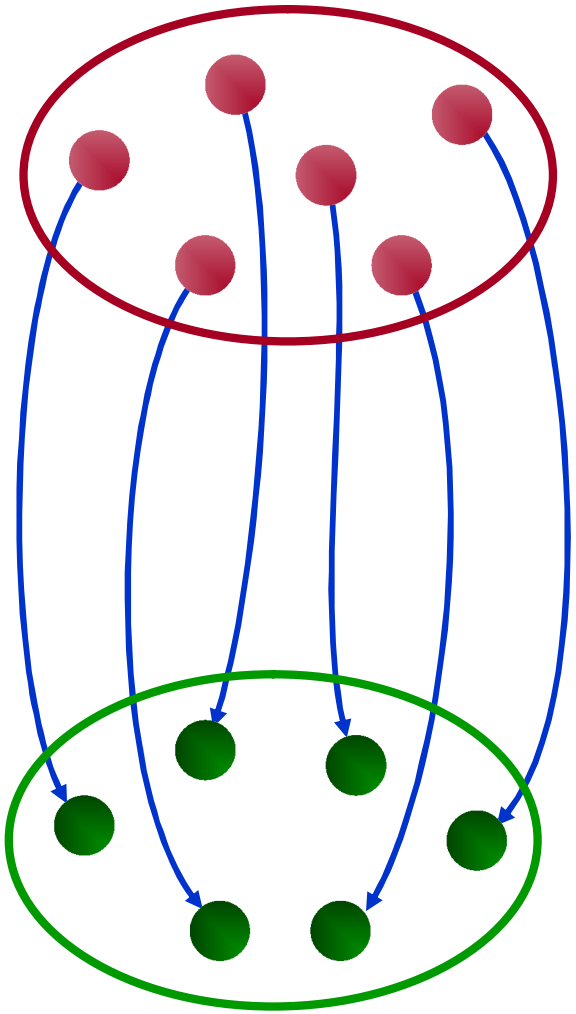
Supporto fisico: 2a condizione

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



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**

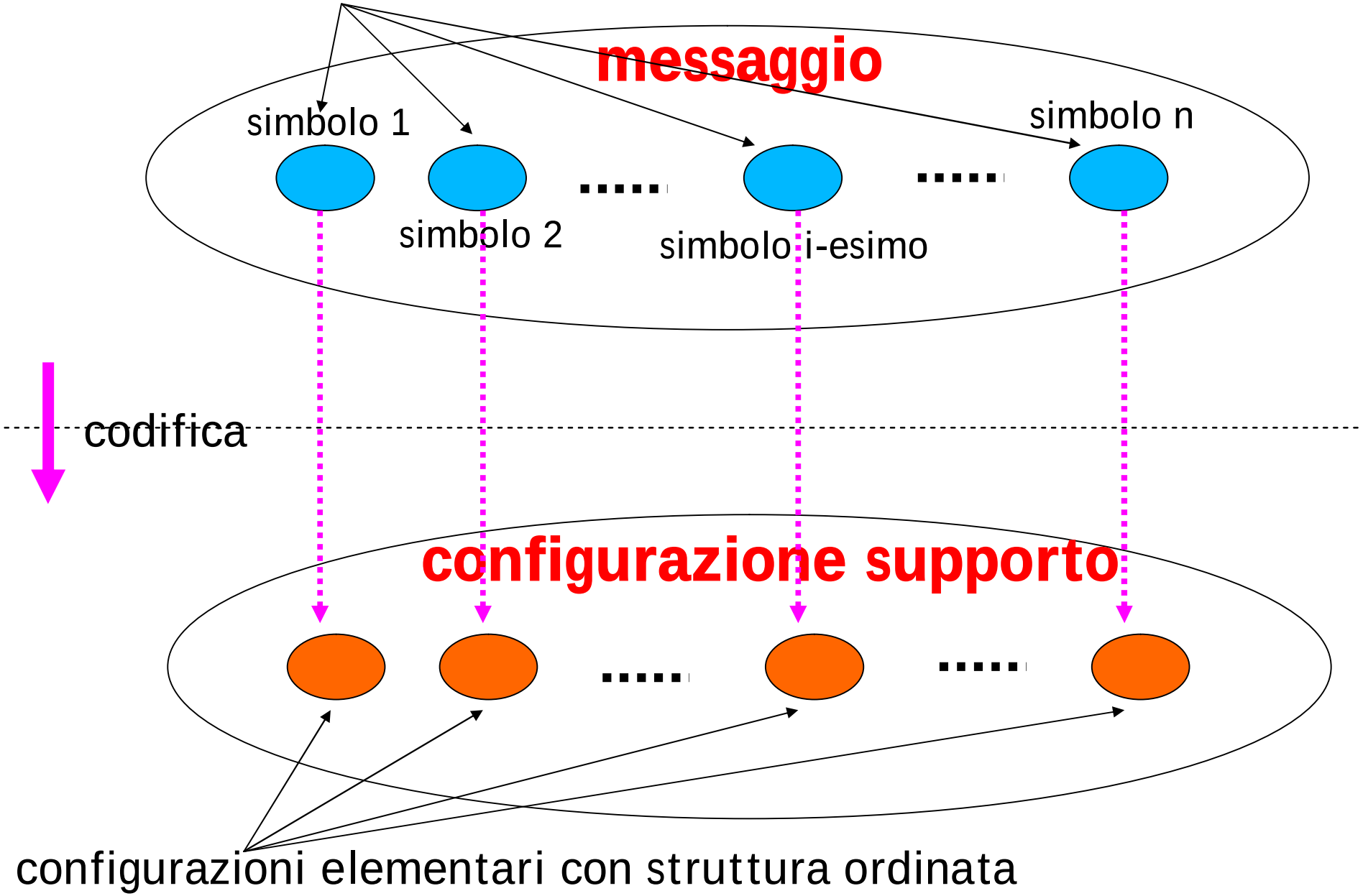
Configurazioni, configurazioni elementari, simboli e messaggi 1/2

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**" = $1\times10^3 + 2\times10^2 + 3\times10^1 + 4\times10^0 + 5\times10^{-1}$
  - "**1,23,45**" = ??
- **Lo stesso alfabeto può essere utilizzato con codici diversi:**
  - "**123,456**" = $1\times10^2 + 2\times10^1 + 3\times10^0 + 4\times10^{-1} + 5\times10^{-2} + 6\times10^{-3}$, [IT]
  - "**123,456**" = $1\times10^5 + 2\times10^4 + 3\times10^3 + 4\times10^2 + 5\times10^1 + 6\times10^0$, [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

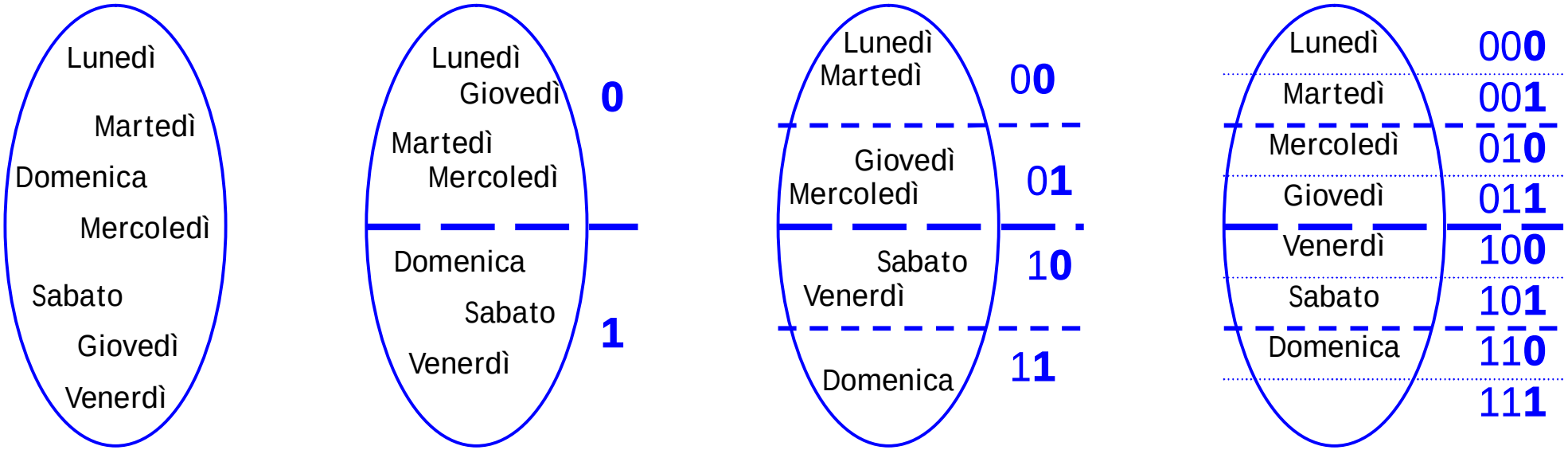
- **Problema: assegnare un codice binario univoco a tutti i giorni della settimana**

- **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)

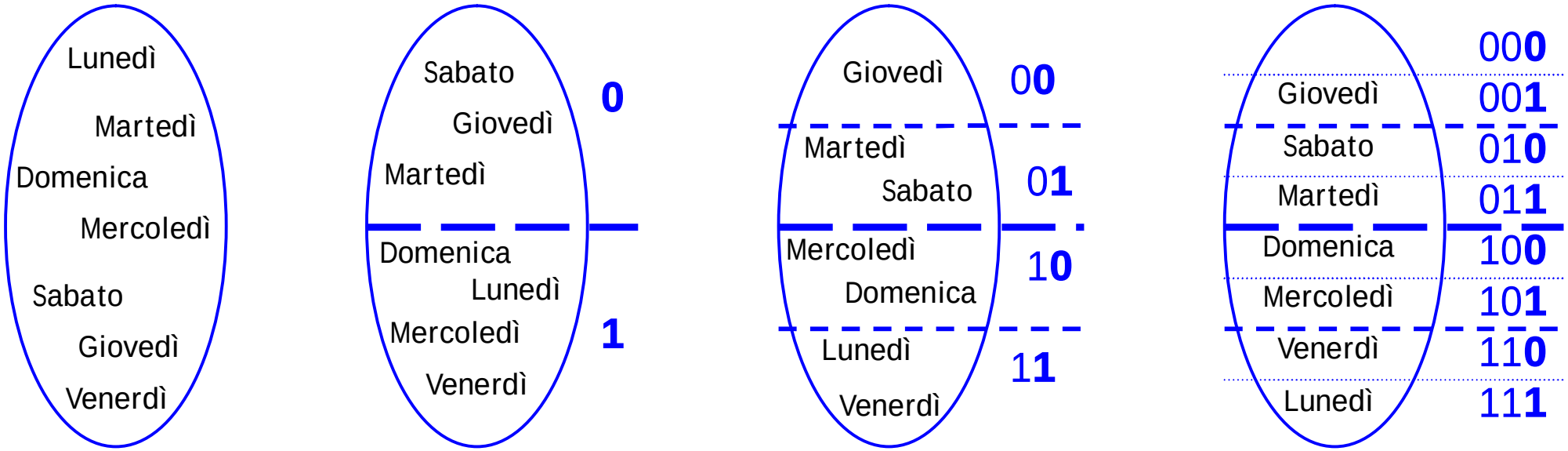


1 bit
2 “gruppi”

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	**ADD**
0111 1101	**SUB**
0111 1110	**AND**
… … …	**… … …**

Istruzioni per il trasferimento dati	
Codice	Istruzione
1110 1000	**LOAD**
1111 1000	**STORE**
… … …	**… … …**
… … …	**… … …**

Istruzioni di controllo	
Codice	Istruzione
0100 1001	**IF_EQ**
0100 1000	**GOTO**
0100 1100	**RETURN**
… … …	**… … …**

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.**

Rappresentazione dei numeri

Rappresentazione dei numeri naturali

La rappresentazione dei numeri naturali (0,1,2,...) 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,...) 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, …0) di N moltiplicate per coefficienti (C_k, C_{k-1},… 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},…C_0 minori di N e con $C_k \neq 0$, tale che:

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

Esempi:

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

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

Esempio 1: Rappresentazione dei numeri naturali

Assumiamo: base N=10 ed M = “duemilacentootto”

- 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 (centootto), 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, duemilacentootto in base 10 è rappresentato da:

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

Esempio 2: Rappresentazione dei numeri naturali

Assumiamo: base N=2 ed M = “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 volte, 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:

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

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**
- $N = c_k c_{k-1} \ldots c_0$ rappresenta $c_k \times b^k + c_{k-1} \times b^{k-1} + \ldots + c_0 \times b^0$
  - b=10 => 1101_{dieci} indica $1 \times 10^3 + 1 \times 10^2 + 0 \times 10 + 1 \times 10^0$
  - b=2 => 1101_{due} indica $1 \times 10^{11} + 1 \times 10^{10} + 0 \times 10^1 + 1 \times 10^0$
- **Conversione binario $\Rightarrow$ decimale**
  - basta scrivere il numero secondo la notazione posizionale utilizzando già il sistema decimale
  - b=2 $\Rightarrow$ 1101_{due} indica $1 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 \times 2^0 = 13_{dieci}$

Conversione binario ⇒ decimale

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

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

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)**

		+ (XOR) Disg. Excl.	OR Disg. Incl.	x	AND Congiunz.
0	0	0	0	0	0
0	1	1	1	0	0
1	0	1	1	0	0
1	1	0	1	1	1

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_{dieci} a 255_{dieci}.
- Operazioni che producono un risultato maggiore provocano il superamento della capacità di rappresentazione (indicato in gergo dal termine inglese overflow).

	1 1				1 1		
	00011001_{due} +	25_{dieci} +			00101100_{due} +	44_{dieci} +	
	00111100_{due} =	60_{dieci} =			01001110_{due} =	78_{dieci} =	
	01010101_{due}	85_{dieci}			01111010_{due}	122_{dieci}	

Numeri binari: operazioni x

- **Operazioni di moltiplicazione dei numeri binari.**
- **Facilitate dal fatto che le moltiplicazioni per una cifra non richiedono calcoli (Nx0=0, Nx1=N)**
- **Le moltiplicazioni a più cifre si effettuano per somme successive (algoritmo manuale)**

Esempio: 0011_{due} X 3_{dieci}

0011_{due} = 3_{dieci}

```
-----------------------
  0011
 0011
0000
0000
-------------
```

0001001_{due} 9_{dieci}

Numeri Naturali: conversione decimale => binario

Si potrebbe utilizzare lo stesso metodo posizionale usato sopra

- B=10 => $345_{dieci} = 3\times10^2 + 4\times10^1 + 5\times10^0$
- B=2 => $345_{dieci} = 011\times1010^{10} + 100\times1010^1 + 101\times1010^0$

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

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

```
  101
```

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

Numeri Naturali: conversione decimale => binario

Esempio: da decimale a binario

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

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

Conversione decimale ⇒ 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

s

$$c_{n-1}c_{n-2}\ldots c_1c_0 = c_{n-1}\times B^{n-1} + c_{n-2}\times B^{n-2} + \ldots + c_1\times B^1 + c_0\times B^0$$
$$c_{n-1}c_{n-2}\ldots c_1c_0 = c_{n-1}\times B^{n-1} + c_{n-2}\times B^{n-2} + \ldots + c_1\times B + c_0 \qquad \text{(infatti } B^1 = B \text{ e } B^0 = 1\text{)}$$

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

$$(c_{n-1}\times B^{n-1} + c_{n-2}\times B^{n-2} + \ldots + c_1\times B + c_0)/B = c_{n-1}\times B^{n-2} + c_{n-2}\times B^{n-3} + \ldots + c_1 + c_0/B$$

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

$$\text{quoziente} = c_{n-1}\times B^{n-2} + c_{n-2}\times B^{n-3} + \ldots + c_1 \qquad \text{resto} = c_0$$

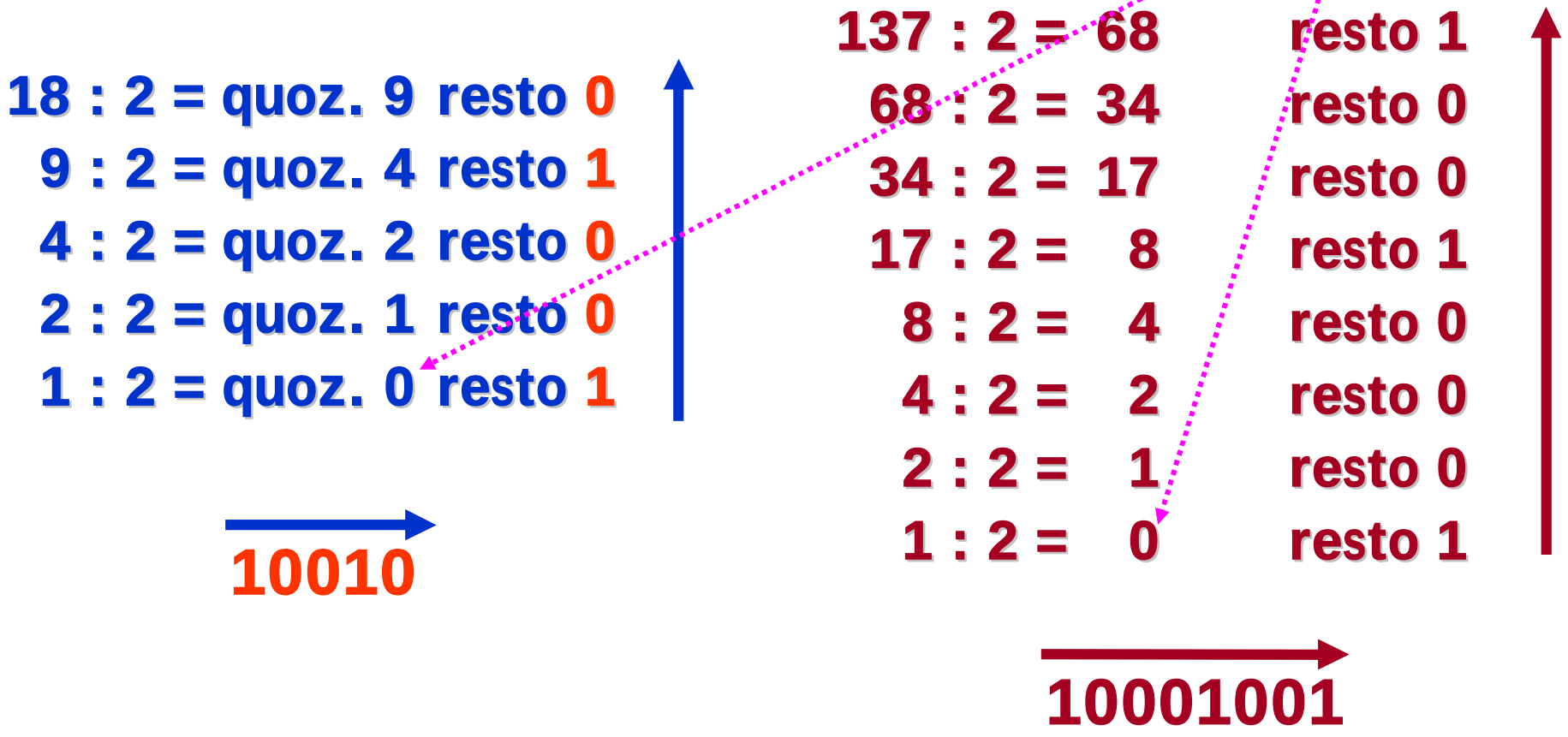
- Il resto della divisione corrisponde all'ultima cifra della rappresentazione in base B del numero, ma il suo valore è indipendente dalla base B che si utilizza per effettuare i conti.
- Applicando lo stesso procedimento al quoziente si ottiene la penultima cifra della rappresentazione in base B
- Iterando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

Conversione decimale ⇒ binario

(metodo “semplice”)

Si calcolano i resti delle divisioni per 2 fino al quoziente 0

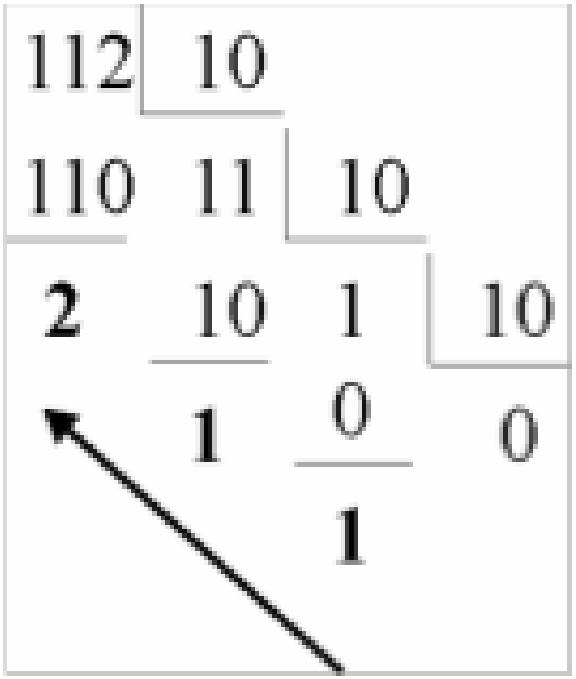
Si considerano i resti nell’ordine dall’ultimo (in basso) al primo (in alto)



Conversione decimale-binario

Perché l'algoritmo della divisione funziona ?

Per scoprire il motivo per cui l'algoritmo funziona, analizziamo cosa succede nel caso on cui, invece di dividere per 2, si divide per 10



La prima divisione per 10 consente di "isolare", cioè stabilire quante sono le unità ($2x10^0$), la seconda le decine ($1x10^1$), la terza le centinaia ($1x10^2$) e così via.

La stessa cosa succede se si divide per 2 invece che per 10. In questo caso, però si isolano le potenze di 2 e non le unità o le decine…

Infatti: $112_{dieci} => 1x10^2 + 1x10^1 + 2x10^0$

Conversione decimale ⇒ binario

(cifra binaria meno significativa)

573_{dieci} : 2_{dieci}	⇒	*quoziente*	286_{dieci}	*resto* 1_{dieci}
286_{dieci} : 2_{dieci}	⇒	*quoziente*	143_{dieci}	*resto* 0_{dieci}
143_{dieci} : 2_{dieci}	⇒	*quoziente*	71_{dieci}	*resto* 1_{dieci}
71_{dieci} : 2_{dieci}	⇒	*quoziente*	35_{dieci}	*resto* 1_{dieci}
35_{dieci} : 2_{dieci}	⇒	*quoziente*	17_{dieci}	*resto* 1_{dieci}
17_{dieci} : 2_{dieci}	⇒	*quoziente*	8_{dieci}	*resto* 1_{dieci}
8_{dieci} : 2_{dieci}	⇒	*quoziente*	4_{dieci}	*resto* 0_{dieci}
4_{dieci} : 2_{dieci}	⇒	*quoziente*	2_{dieci}	*resto* 0_{dieci}
2_{dieci} : 2_{dieci}	⇒	*quoziente*	1_{dieci}	*resto* 0_{dieci}
1_{dieci} : 2_{dieci}	⇒	*quoziente*	0_{dieci}	*resto* 1_{dieci}

(cifra binaria più significativa)

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

Codifica ed ambiguità

- 10_{due} => 2_{dieci}
- 010_{due} => 2_{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 ≈ 4×10^{9}.

- Raddoppiando la lunghezza delle successioni il massimo numero rappresentabile cresce esponenzialmente: passando da 32 a 64 bit il massimo numero rappresentabile diventa $2^{64}-1 \approx 16\times10^{18} = 1.6\times10^{19}$ (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:**

236: 8 quoz. 29 resto 4
29: 8 quoz. 3 resto 5
3: 8 quoz. 0 resto 3

Lettura (posizionale) Ottale:

354_{otto} => $3\times10^2 + 5\times10^1 + 4\times10^0$ = 300+50+4 = 354_{otto}

- **Trasformazione da Ottale a Decimale:**

354_{otto} => $3\times8^2 + 5\times8^1 + 4\times8^0$ = 192+40+4 = 236_{dieci}

1461_{otto} => $1\times8^3 + 4\times8^2 + 6\times8^1 + 1\times8^0$ = 512+256+48+1 = 817_{dieci}

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

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

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

Esadecimali

- **Per rappresentare sinteticamente i valori binari**
- **Esadecimali (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:

  $EC_{sedici} = 14\times16^1 + 12\times16^0 = 224 + 12 = 236_{dieci}$

  $331_{sedici} = 3\times16^2 + 3\times16^1 + 1\times16^0 = 768+48+1 = 817_{dieci}$

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

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

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

Numeri interi (+/-)

- **Alfabeto binario**
  - il segno è rappresentato da **0** (**+**) oppure **1** (**-**)
  - è indispensabile indicare il numero **k** di bit utilizzati
- **Modulo e segno**
  - **1** bit di segno (**0 positivo**, **1 negativo**)
  - **k – 1** bit di modulo
    - **Esempio:** $+6_{dieci} = 0110_{ms}$ $-6_{dieci} = 1110_{ms}$
  - si rappresentano i valori da $-2^{k-1}+1$ a $2^{k-1}-1$
    - con **4 bit** i valori vanno da $-7 (= -2^{4-1}+1)$ a $+7 (= 2^{4-1}-1)$
    - con **8 bit** i valori vanno da $-127 (= -2^{8-1}+1)$ a $+127 (= 2^{8-1}-1)$

Problema: ci sono due rappresentazioni dello 0 (ambiguità!)

con 4 bit avremo: $+0_{dieci} = 0000_{ms}$ e $-0_{dieci} = 1000_{ms}$

Diverse codifiche/interpretazioni

Codice	Nat	MS
0000	0	**+0**
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	**-0**
1001	9	-1
1010	10	-2
1011	11	-3
1100	12	-4
1101	13	-5
1110	14	-6
1111	15	-7

Numeri interi in “Complemento a 2”

Il complemento a due è il metodo più diffuso per la rappresentazione dei numeri interi (+/-) in informatica.

E’ stato inventato intorno al 1960 per evitare la presenza di “due zeri” (uno positivo e uno negativo) e per utilizzare tutte le possibili 2^n combinazioni che si possono ottenere con n bit.

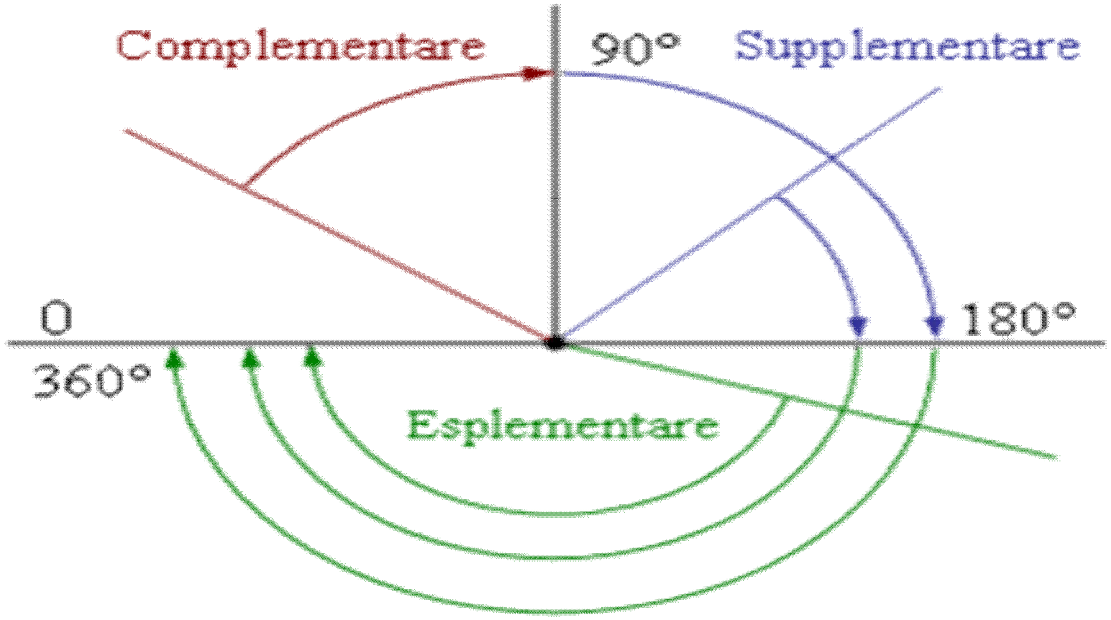
Il bit iniziale (più a sinistra) del numero indica il segno

+ (0) - (1)

mentre la parte restante indica il modulo

Il complemento in mat e geo

In geometria: si dice ***complementare*** di un angolo di ampiezza α ogni angolo avente l'ampiezza la ß "mancante" per ottenere un angolo retto:



In matematica: il numero X che manca a un altro numero Y per ottenere un terzo numero Z è detto "**complemento a Z di Y"**:
Esempio: il complemento a 7 di 4 é 3 (essendo 4+3=7) o, in altre parole, "**4 complemento a 7 è uguale a 3"**.

Complemento a 1 e Complemento a 2

C_1: Per complemento a 1 di un numero binario X, si intende quel numero (binario) Y che, sommato al numero (binario) di partenza X dia 1 come risultato:

per ottenere Y basta invertire o negare i bit di X

C_2: Per complemento a 2 di un numero binario X, si intende quel numero (binario) Y che, sommato al numero (binario) di partenza X dia 0 come risultato:

per ottenere Y basta aggiungere 1 al Complemento a 1 di X

Numeri interi in Complemento a 2

Esempio: rappresentiamo -5 con 4 bit in "Complemento a 2".
HINT: possiamo vedere -5 come il Complemento a 2 di +5

Si scrive innanzitutto la rappresentazione binaria del numero +5:

$$+5_{10} = 0101_2$$

Si scrive il "**complemento a 1**" del binario così ottenuto
si invertono cioè i bit (0 diventa 1 e 1 diventa 0), quindi:

$$1010_{C1}$$

Per ottenere il "**complemento a due**" di +5 aggiungiamo 1 al C1:

$$1011_{C2} = -5$$

Osserva: $(0101)_{C2} + (1011)_{C2} = 10000$ (il primo bit a sx è cancellato)

complemento a 2: un altro esempio

calcola la rappresentazione di –X a partire da quella di X e viceversa

Dato X procedere al complemento di ogni bit di X (cioè il C1 di X) , poi aggiungere 1 (cioè il C2 di X)

- rappresentazione di $+6_{dieci} = 0110_{C2}$
  (NB ci vogliono 4 bit!!)
- Complemento (a 1) di tutti i bit $\Rightarrow 1001_{C2}$
  (corrisponderebbe a -7_{dieci})
- aggiungere 1 $\Rightarrow 1010_{C2}$
  (che corrisponde a -6_{dieci})

Numeri interi in *Complemento a 2*

- **Alfabeto binario**
  - il segno è rappresentato sempre da 0 (+) o 1 (-)
  - è indispensabile indicare il numero **k** di bit utilizzati
- **Complemento a 2 (metodo diretto)**
  - **X** corrisponde alla versione binaria del naturale $2^k + X$

    $+6_{dieci} \Rightarrow 2^4+6 = 22 \Rightarrow [1]0110 \Rightarrow 0110_{C2}$ **(cancelliamo [1])**

    $-6_{dieci} \Rightarrow 2^4- 6 = 10 \Rightarrow [0]1010 \Rightarrow 1010_{C2}$ **(cancelliamo [0])**
  - si rappresentano i valori che vanno da -2^{k-1} a $2^{k-1}-1$
    - con 4 bit i valori vanno da –8 a +7
    - con 8 bit i valori vanno da –128 a +127
    - Con 32 bit i valori vanno da –2'147'483'648 fino a +2'147'483'647
  - **Attenzione: c'è una sola rappresentazione dello 0**
    - con 4 bit : $+0_{dieci} = 0000_{C2}$ mentre $1000_{C2} = -8_{dieci}$

Il complemento a 2

Ancora un altro metodo indiretto per calcolare

la rappresentazione di -X a partire da quella di X e viceversa

Partendo da destra e andando verso sinistra, si lasciano invariati tutti i bit fino al primo 1 compreso, poi si complementano tutti gli altri bit.

Esempio:

- rappresentazione di $+6_{dieci}$ = $\mathbf{0110_{C2}}$ (NB ci vogliono 4 bit!!)
- gli ultimi due bit (_ _ **1 0**) rimangono invariati
- gli altri due bit vengono complementati:

$\mathbf{1010_{C2}}$ che corrisponde a -6_{dieci}

Complemento a 2 => Decimale

- Se si considera la notazione posizionale, si può notare che nella rappresentazione binaria in complemento a due il valore (decimale) si può ottenere associando alla cifra più significativa (a sinistra) un peso negativo mentre tutte le altre cifre mantengono il peso originario positivo.

- **Il valore decimale di un numero $c_{n-1}c_{n-2}...c_1c_0$ scritto in complemento a due è :**

$$-c_{n-1}\times 2^{n-1} + c_{n-2}\times 2^{n-2} + \ldots + c_1\times 2^1 + c_0\times 2^0$$

- **Esempi:**
  - $0101_{C2} = -0\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0 = +5_{dieci}$
  - $1011_{C2} = -1\times 2^3+0\times 2^2+1\times 2^1+1\times 2^0 = -5_{dieci}$

Complemento a 2: alcune osservazioni

- **I valori positivi iniziano con 0, quelli negativi con 1**
- **Data la rappresentazione di un numero su k bit, la rappresentazione dello stesso numero su k+1 bit si ottiene aggiungendo (a sinistra) un bit uguale al primo ("estensione del segno")**
  - Rappresentazione di –6 su 4 bit = **1010**
  - Rappresentazione di –6 su 5 bit = **11010**
  - Rappresentazione di –6 su 8 bit = **11111010**
- **la sottrazione si effettua come somma algebrica**
  - 4 – 6 = +4 + (–6) = 0100 + 1010 = 1110 = –2
  - 9 – 6 = +9 + (–6) = 01001 + 11010 = [1]00011 = +3

C2: rappresentazioni con diversi #bit

Naturale in base dieci	Complemento a due (# bit)					
	2	3	4	5	8	16
–13	NA	NA	NA	1 0011	1111 0011	1111 1111 1111 0011
–8	NA	NA	1000	1 1000	1111 1000	1111 1111 1111 1000
–7	NA	NA	1001	1 1001	1111 1001	1111 1111 1111 1001
–4	NA	100	1100	1 1100	1111 1100	1111 1111 1111 1100
–1	11	111	1111	1 1111	1111 1111	1111 1111 1111 1111
+1	01	001	0001	0 0001	0000 0001	0000 0000 0000 0001
+4	NA	NA	0100	0 0100	0000 0100	0000 0000 0000 0100
+7	NA	NA	0111	0 0111	0000 0111	0000 0000 0000 0111
+8	NA	NA	NA	0 1000	0000 1000	0000 0000 0000 1000
+13	NA	NA	NA	0 1101	0000 1101	0000 0000 0000 1101

C2: operazioni

- Operazioni di somma di numeri binari in complemento a 2
- Con gli 8 bit utilizzati negli esempi qui riportati si possono rappresentare i numeri interi da -128_{dieci} fino a $+127_{dieci}$

$$\begin{array}{rr} {\scriptstyle 1\ 1} & \\ 00011001_{C2} + & +25_{dieci} + \\ 00111100_{C2} = & +60_{dieci} = \\ \hline 01010101_{C2} & +85_{dieci} \end{array}$$

$$\begin{array}{rr} {\scriptstyle 1} & \\ 00101100_{C2} + & +44_{dieci} + \\ 10110010_{C2} = & -78_{dieci} = \\ \hline 11011110_{C2} & -34_{dieci} \end{array}$$

$$\begin{array}{rr} {\scriptstyle 1\ 1\quad\ 1} & \\ 11100111_{C2} + & -25_{dieci} + \\ 11000100_{C2} = & -60_{dieci} = \\ \hline \cancel{1}10101011_{C2} & -85_{dieci} \end{array}$$

$$\begin{array}{rr} {\scriptstyle 1\ 1\ \ 1\ 1\ 1} & \\ 11010100_{C2} + & -44_{dieci} + \\ 01001110_{C2} = & +78_{dieci} = \\ \hline \cancel{1}00100010_{C2} & +34_{dieci} \end{array}$$

C2: operazioni e overflow

Operazioni di somma di numeri binari in complemento a 2
Con gli 8 bit utilizzati negli esempi qui riportati si possono rappresentare i numeri interi da -128_{dieci} fino a $+127_{dieci}$

- **Se due operandi dello stesso segno danno un risultato di segno opposto vuol dire che è stata superata la capacità di calcolo** (*overflow*).
- Banalmente, con due operandi di segno opposto l'overflow non può mai verificarsi

Diverse codifiche/interpretazioni

Dieci	Due	MS	C2
+0	0000	0000	0000
+1	0001	0001	0001
+2	0010	0010	0010
+3	0011	0011	0011
+4	0100	0100	0100
+5	0101	0101	0101
+6	0110	0110	0110
+7	0111	0111	0111
+8	1000	NA	NA

Dieci	Due	MS	C2
–0	0000	1000	0000
–1	NA	1001	1111
–2	NA	1010	1110
–3	NA	1011	1101
–4	NA	1100	1100
–5	NA	1101	1011
–6	NA	1110	1010
–7	NA	1111	1001
–8	NA	NA	1000

Diverse codifiche/interpretazioni

Codice	Nat	MS	C2
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7

Codice	Nat	MS	C2
1000	8	-0	-8
1001	9	-1	-7
1010	10	-2	-6
1011	11	-3	-5
1100	12	-4	-4
1101	13	-5	-3
1110	14	-6	-2
1111	15	-7	-1

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.

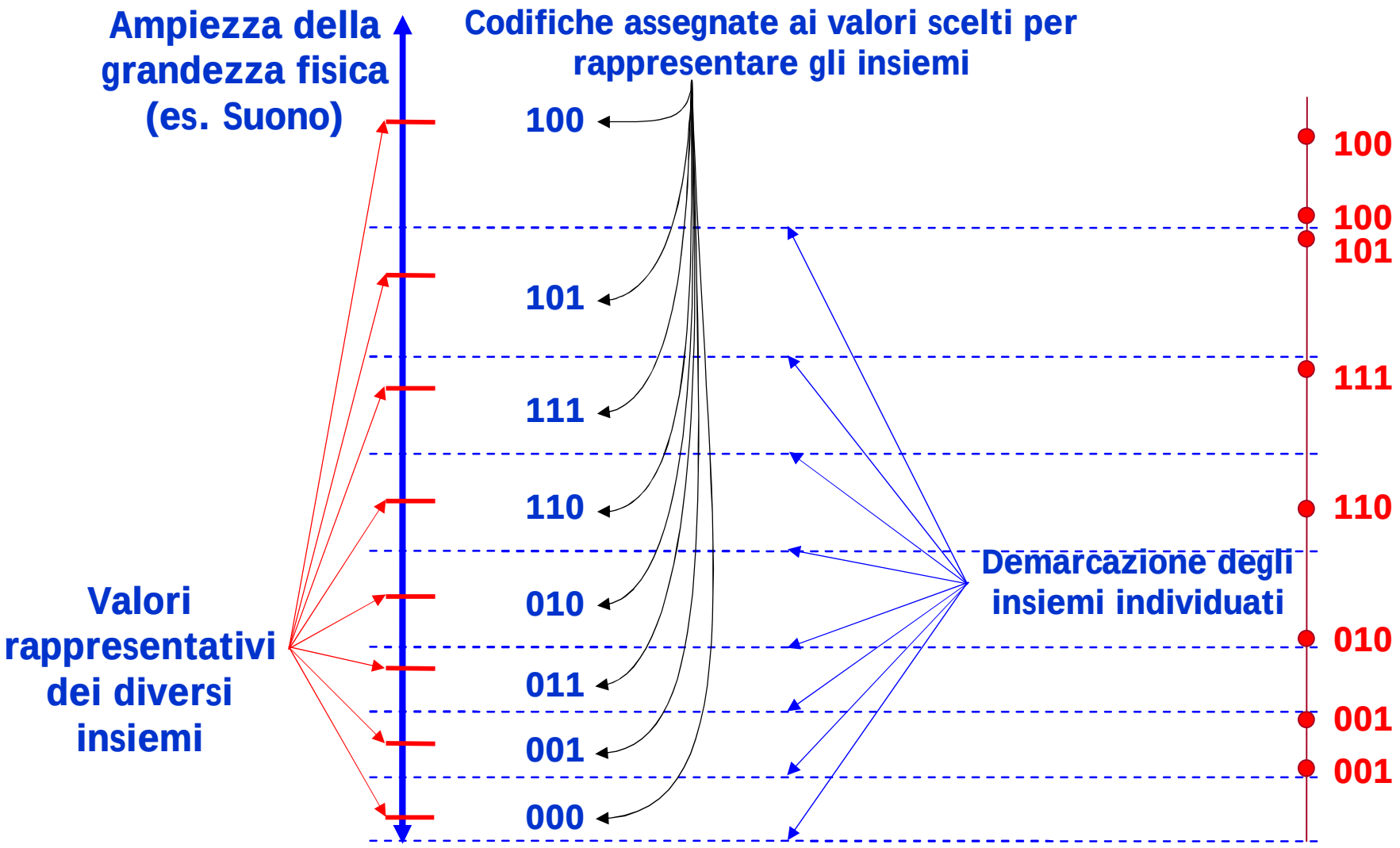
- **Cosa succede nei due casi se si aggiungono i "mezzi punti"?**

Analogico e digitale

- **Meta-informazione esplicita nel supporto**
- **Codifica ANALOGICA**

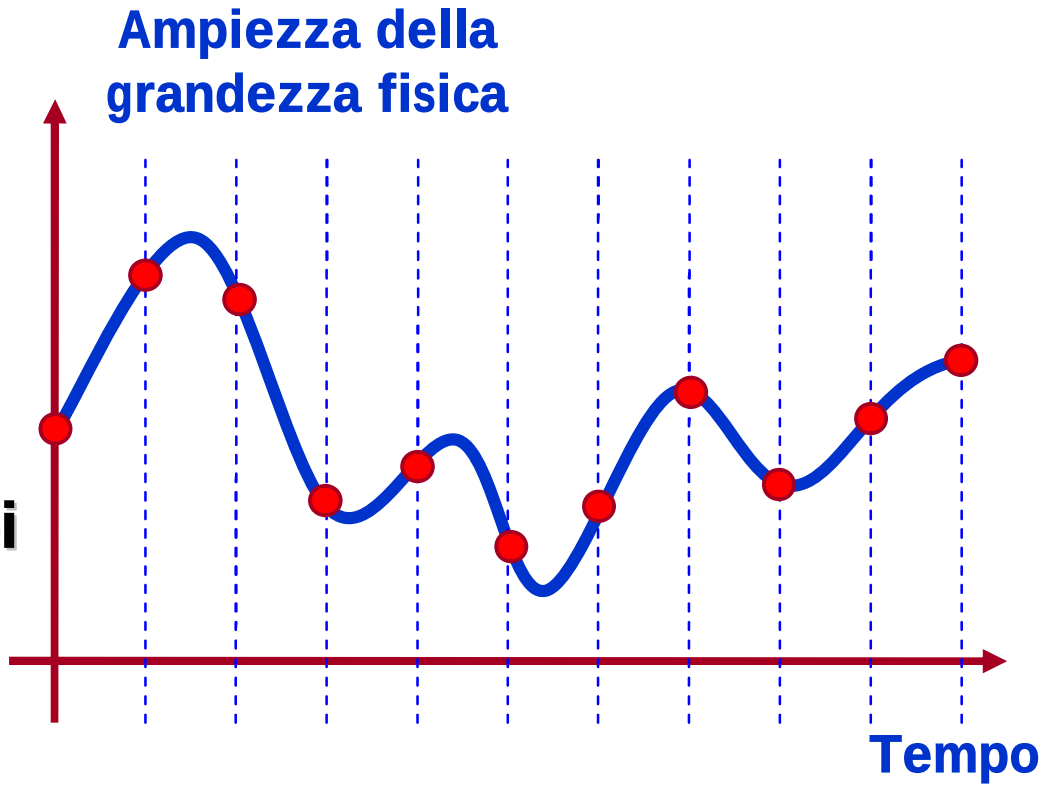
- **Meta-informazione implicita nella codifica**
- **Codifica DIGITALE**

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.



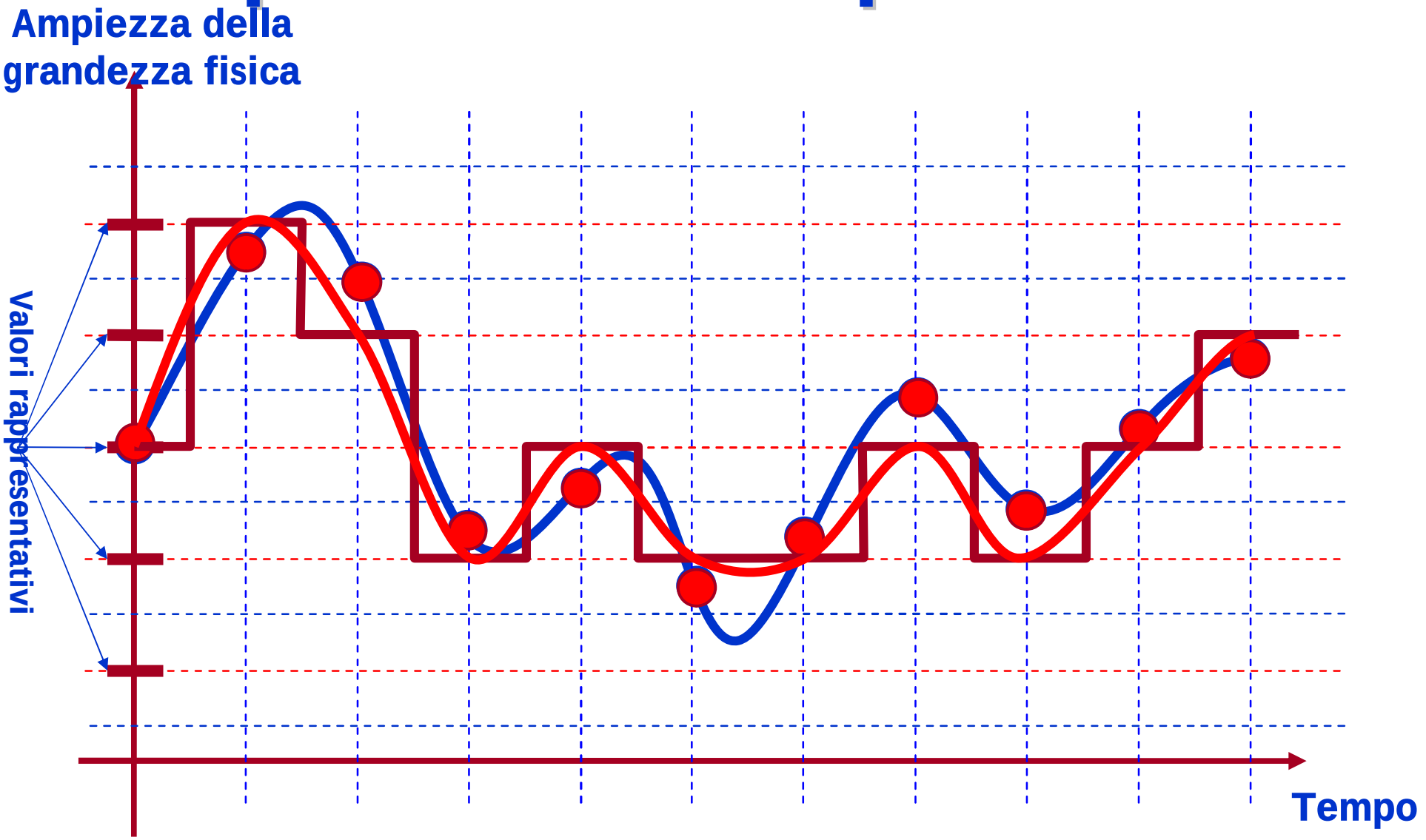
Da analogico a digitale: quantizzazione e campionamento

- il parametro che caratterizza la quantizzazione (suddivisione di una grandezza fisica in intervalli) è il numero di BIT

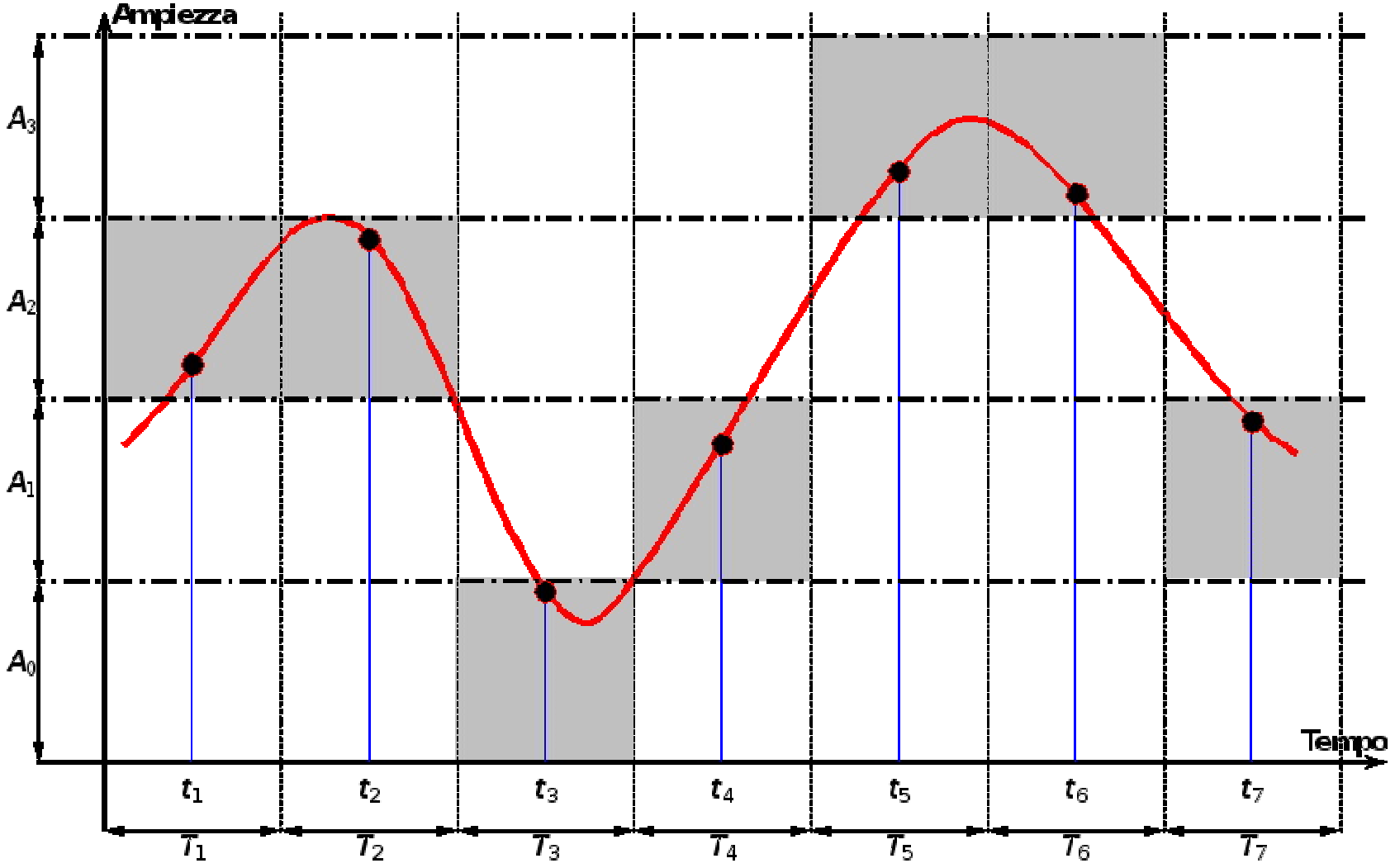
- il parametro che caratterizza il campionamento è la frequenza di campionamento, cioè il numero di campioni che vengono acquisiti nell'unità di tempo e si misura in Hz (Hertz) al secondo.

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

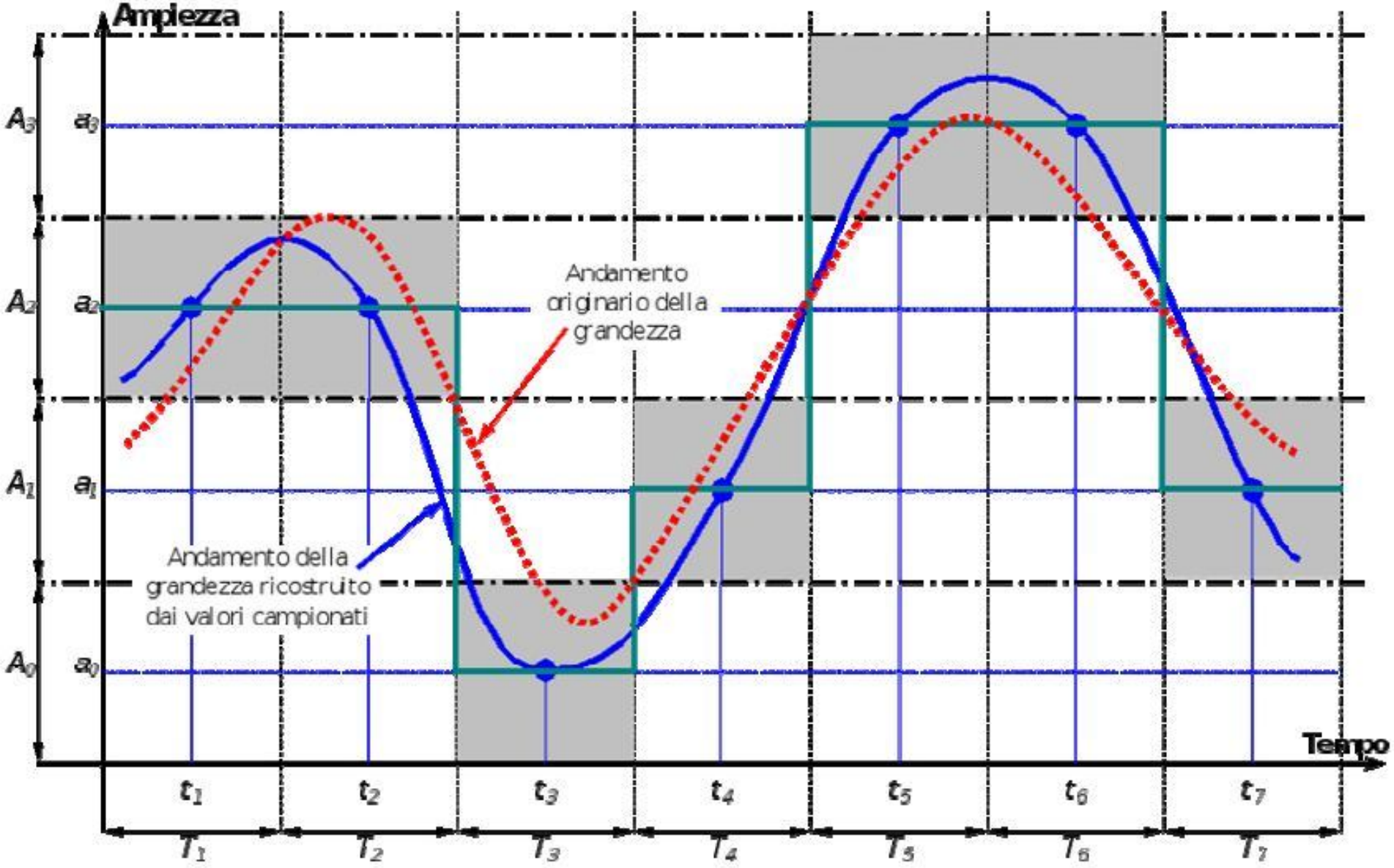
Campionamento e quantizzazione



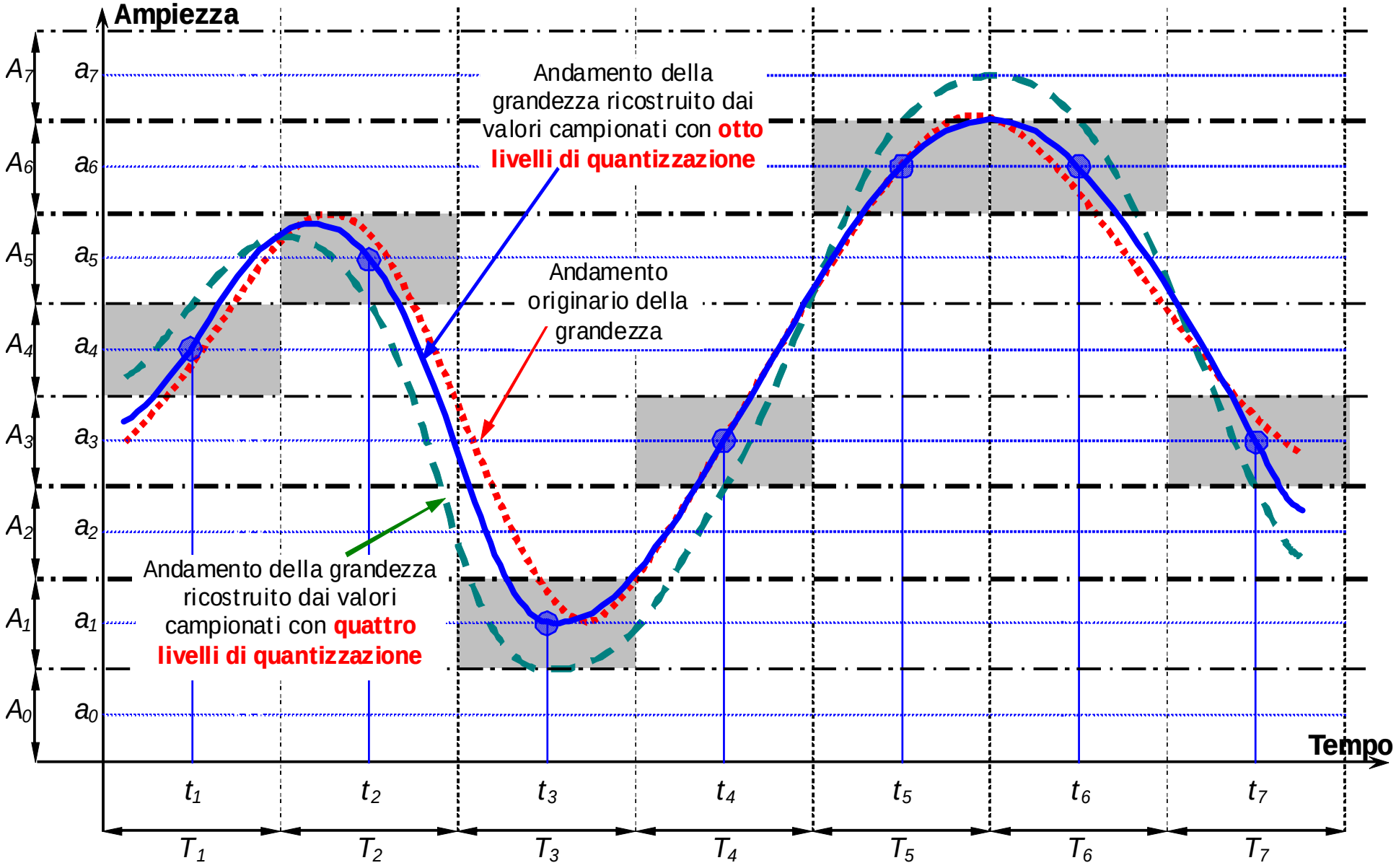
Campionamento a 2 BIT (4 configurazioni) e Quantizzazione a 7 Hertz (7 campioni al secondo)



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 x 2 bytes x 2 canali).

- L'errore che si commette nella ricostruzione del segnale sonoro è difficilmente rilevabile da parte di un orecchio umano.

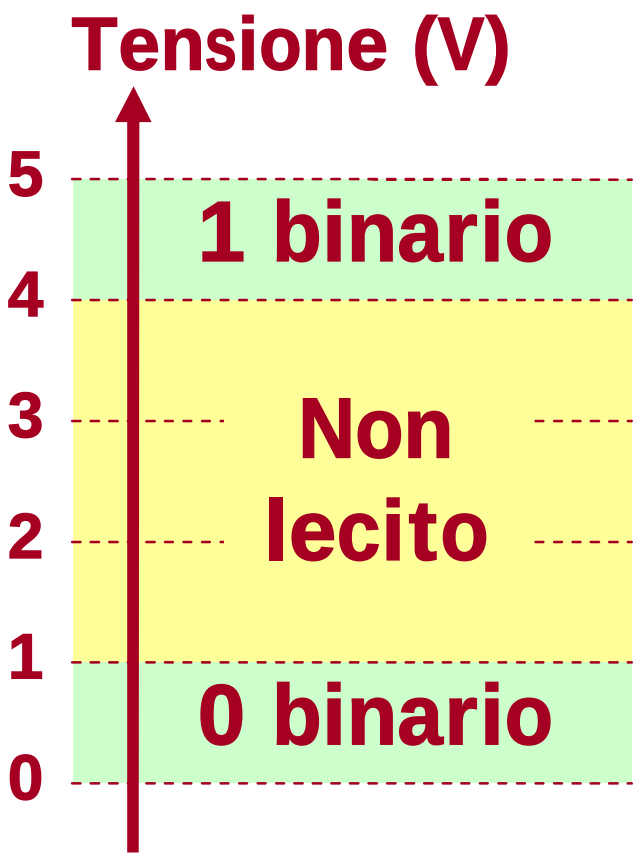
Foto digitali

- **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**
    (1×50% + 2×25% + 3×12.5% + 3×12.5%) bit/carattere × 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