Codifiche di carattere

Un carattere è un simbolo appartenente a una stringa testuale:

- cifre, lettere, simboli di punteggiatura

Codifiche di carattere

Un carattere è un simbolo appartenente a una stringa testuale:

- cifre, lettere, simboli di punteggiatura
- simboli speciali: ‘@’, ‘#’, ‘$’, ‘%’, ‘&’, ‘)’, ‘ ‘ ’, . . .

Codifiche di carattere

Un carattere è un simbolo appartenente a una stringa testuale:

- cifre, lettere, simboli di punteggiatura
- simboli speciali: ‘@’, ‘#’, ‘$’, ‘%’, ‘&’, ‘)’, ‘ ‘ ’, . . .
- caratteri speciali: contengono informazioni di controllo, definiscono il formato del testo, impartiscono comandi come ritorno a capo, tabulazione, *escape* (ESC), spostamento del cursore (prompt), ...

Codici per caratteri

Il codice C può essere arbitrario, con alcune utili regole:

- cifre consecutive mappate su codifiche consecutive. Es.: $C(\text{‘N’}) = C(\text{‘0’}) + N$
- lettere consecutive mappate su codifiche consecutive. Es.:
  $C(\text{‘C’}) = C(\text{‘A’})$ + posizione lettera ‘C’ nell’alfabeto - 1.

Principali codici: ASCII (standard 8 bit ed esteso), MS DOS, MAC OS Roman, UNICODE, UTF-8, UTF-7, UTF-16, EBCDIC,

Codici per caratteri

Il codice C può essere arbitrario, con alcune utili regole:

- cifre consecutive mappate su codifiche consecutive. Es.: $C(\text{‘N’}) = C(\text{‘0’}) + N$
- lettere consecutive mappate su codifiche consecutive. Es.:
  $C(\text{‘C’}) = C(\text{‘A’})$ + posizione lettera ‘C’ nell’alfabeto - 1.

Principali codici: ASCII (standard 8 bit ed esteso), MS DOS, MAC OS Roman, UNICODE, UTF-8, UTF-7, UTF-16, EBCDIC, Morse.

Codice ASCII

American Standard Code for Information Interchange

Prima codifica condivisa e a larga diffusione (anni '60), 7 bit per carattere:

- codici da 0 a 31 dedicati al controllo del testo (carriage return, line feed, backspace, cancel, escape, ...) e del flusso da/a terminale (start of heading, end of transmission, ...)

Codice ASCII

American Standard Code for Information Interchange

Prima codifica condivisa e a larga diffusione (anni '60), 7 bit per carattere:

- codici da 0 a 31 dedicati al controllo del testo (carriage return, line feed, backspace, cancel, escape, ...) e del flusso da/a terminale (start of heading, end of transmission, ...)
- codici da 32 a 126 dedicati a 94 caratteri stampabili

Codice ASCII

American Standard Code for Information Interchange

Prima codifica condivisa e a larga diffusione (anni '60), 7 bit per carattere:

- codici da 0 a 31 dedicati al controllo del testo (carriage return, line feed, backspace, cancel, escape, ...) e del flusso da/a terminale (start of heading, end of transmission, ...)
- codici da 32 a 126 dedicati a 94 caratteri stampabili
- 127: delete.

Codice ASCII

Hex	Name	Meaning	Hex	Name	Meaning
0	NUL	Null	10	DLE	Data Link Escape
1	SOH	Start Of Heading	11	DC1	Device Control 1
2	STX	Start Of TeXt	12	DC2	Device Control 2
3	ETX	End Of TeXt	13	DC3	Device Control 3
4	EOT	End Of Transmission	14	DC4	Device Control 4
5	ENQ	Enquiry	15	NAK	Negative AcKnowledgement
6	ACK	ACKnowledgement	16	SYN	SYNchronous idle
7	BEL	BELl	17	ETB	End of Transmission Block
8	BS	BackSpace	18	CAN	CANcel
9	HT	Horizontal Tab	19	EM	End of Medium
A	LF	Line Feed	1A	SUB	SUBstitute
B	VT	Vertical Tab	1B	ESC	ESCape
C	FF	Form Feed	1C	FS	File Separator
D	CR	Carriage Return	1D	GS	Group Separator
E	SO	Shift Out	1E	RS	Record Separator
F	SI	Shift In	1F	US	Unit Separator

Codice ASCII

Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char	Hex	Char
20	(Space)	30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	’	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

Codice ASCII - Controllo

I caratteri di controllo erano pensati per la comunicazione tra i terminali e il mainframe (anni '60). Es.: UNIX TTY ("teletype"). Il protocollo è ancora supportato nativamente da Linux (xterm, bash, ...) ma ha limiti intrinseci:

- sistemi operativi diversi gestiscono in diverso modo il ritorno a capo
  ritorno a capo = carriage return (CR) + line feed (LF)
- numero di caratteri rappresentabili largamente insufficiente per una comunicazione di tipo globale.

Estensioni ASCII - IS 8859

Le estensioni standard del codice ASCII (UNIX ANSI, MS-DOS, MAC OS Roman, . . .) conservano le prime 128 codifiche a fini di retrocompatibilità.

Lo standard 8859 classifica le diverse estensioni definendo una code page per ciascuna di esse. Es.:
IS 8859-1: ANSI, Latin 1, West Europe, IS 646
IS 8859-2: Latin 2, East Europe, lingue slave
IS 8859-3: . . .

Il sistema deve sapere su che pagina opera.

Ogni code page non deve contenere caratteri distinti provenienti da idiomi diversi (es.: cinese e giapponese). Alcune code page (stesso esempio) crescono dinamicamente dati i moltissimi caratteri.

UNICODE e UCS

A fronte di una stima globale di oltre 200.000 caratteri, UNICODE rende definitive alcune proprietà proposte in IS 8859

- la lunghezza variabile: due o più byte per carattere
- il code point (codice di carattere) non univoco: per semplicità di traduzione tra code page possono esistere più code point per lo stesso carattere
- code point lasciati vuoti per future estensioni.

UNICODE ha permesso di definire Universal Character Set (UCS).

Codici UCS: UTF

UTF (UCS Transformation Format) è la codifica di caratteri più diffusa, nata per rappresentare UCS in forma compatta: da 1 a 6 byte per carattere.

- UTF-8 (8 bit): 127 caratteri (ASCII standard)
- UTF-16 (16 bit): $\sim 2^{11}$ caratteri: estensioni ASCII, alcuni ideogrammi
- UTF-...: ideogrammi cinesi, altre lingue.

Bits	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	0ddddddd					
11	110ddddd	10dddddd				
16	1110dddd	10dddddd	10dddddd			
21	11110ddd	10dddddd	10dddddd	10dddddd		
26	111110dd	10dddddd	10dddddd	10dddddd	10dddddd	
31	1111110x	10dddddd	10dddddd	10dddddd	10dddddd	10dddddd

Coesistenza di più codici di carattere

L'esistenza di documenti testuali in diversi formati è irreversibile.

I file con contenuto testuale costituiscono la maggioranza dei documenti che vengono scambiati tra computer.

Da molti anni ogni file di testo è provvisto di un header (preambolo) contenente, tra l'altro, l'informazione sul codice di carattere che adopera.

Raramente il codice è presente nel file testo (embedding); se sì, ciò solleva il computer ricevente dal conoscere il codice a priori.

L'ignoranza del codice adoperato produce errori di decodifica del testo ricevuto.

Proprietà di un codice vs. errori

Un codice dev'essere

- compatto: limitare il numero di bit necessari

Proprietà di un codice vs. errori

Un codice dev'essere

- compatto: limitare il numero di bit necessari
- pratico: ottenere le codifiche con calcoli semplici

Proprietà di un codice vs. errori

Un codice dev'essere

- compatto: limitare il numero di bit necessari
- pratico: ottenere le codifiche con calcoli semplici
- accurato: conservare tutta l'informazione, o perderne in quantità trascurabile.

Per contro, nella trasmissione e memorizzazione dei dati si verificano degli errori causati da

- disturbi sulla linea: rumore di fondo e fenomeni elettromagnetici
- imperfezioni nel supporto di memorizzazione
- alterazioni dello stato della memoria per radioattività e fotoattività (DRAM).

Codici di correzione degli errori

Codici di correzione: codici per rilevare ed eventualmente correggere errori presenti nei dati.

Rispondono alla necessità di protezione dagli errori.

Idea base: introdurre ridondanza nell'informazione:

- il trasmettitore/scrittore aggiunge al dato dell'informazione di controllo
- il canale/supporto trasmette/memorizza più bit di quelli strettamente necessari per il dato
- il ricevitore/lettore controlla la presenza di errori sfruttando l'informazione ridondante, che viene rimossa dal dato.

La correzione rende rari ma non impossibili gli errori nell'hardware.

Esempio: il linguaggio parlato

La comunicazione vocale è affetta da errore. Nel linguaggio parlato dunque c'è più informazione di quella strettamente necessaria.

La conoscenza di una parola e del suo contesto permettono di correggere eventuali errori nella comprensione:

- "*oddimo*" viene subito corretto con "*ottimo*"
- "*testo*" può essere confuso con "*desto*", ma "*quali sono i libri di desto*" viene subito corretto in "*quali sono i libri di testo*".

Se la comunicazione è critica allora per correggere un carattere si comunica un'intera parola nota al destinatario: "D come Domodossola!".

Rilevazione e correzione

Un semplice codice trasmette un testo ripetendo ogni carattere due volte: `casa` $\Rightarrow$ `ccaassaa`

Posso in generale rilevare l'errore:

`ccaasraa` $\Rightarrow$ `casa?` `cara?`

Rilevazione e correzione

Un semplice codice trasmette un testo ripetendo ogni carattere due volte: `casa` $\Rightarrow$ `ccaassaa`

Posso in generale rilevare l'errore:

`ccaasraa` $\Rightarrow$ `casa?` `cara?`

Trasmetto un testo ripetendo ogni carattere tre volte.
Posso in generale correggere l'errore:

`cccaaasrsaaa` $\Rightarrow$ `casa`

`cccaaasrraaa` $\Rightarrow$ `cara`

`cccaaasrvaaa` $\Rightarrow$ `casa?` `cara?` `cava?`

Rilevazione e correzione

Un semplice codice trasmette un testo ripetendo ogni carattere due volte: `casa` $\Rightarrow$ `ccaassaa`

Posso in generale rilevare l'errore:

`ccaasraa` $\Rightarrow$ `casa?` `cara?`

Trasmetto un testo ripetendo ogni carattere tre volte.
Posso in generale correggere l'errore:

`cccaaasrsaaa` $\Rightarrow$ `casa`

`cccaaasrraaa` $\Rightarrow$ `cara`

`cccaaasrvaaa` $\Rightarrow$ `casa?` `cara?` `cava?`

La correzione è più forte del rilevamento e come tale richiede codifiche maggiormente ridondanti.

Rilevazione e correzione

Un semplice codice trasmette un testo ripetendo ogni carattere due volte: `casa` $\Rightarrow$ `ccaassaa`

Posso in generale rilevare l'errore:

`ccaasraa` $\Rightarrow$ `casa?` `cara?`

Trasmetto un testo ripetendo ogni carattere tre volte.
Posso in generale correggere l'errore:

`cccaaasrsaaa` $\Rightarrow$ `casa`

`cccaaasrraaa` $\Rightarrow$ `cara`

`cccaaasrvaaa` $\Rightarrow$ `casa?` `cara?` `cava?`

La correzione è più forte del rilevamento e come tale richiede codifiche maggiormente ridondanti.

NON ci sono codici a prova d'errore.

Codici di parità

I codici di parità hanno un singolo bit ridondante. Mettono in primo luogo la compattezza. Sono molto usati nella pratica:

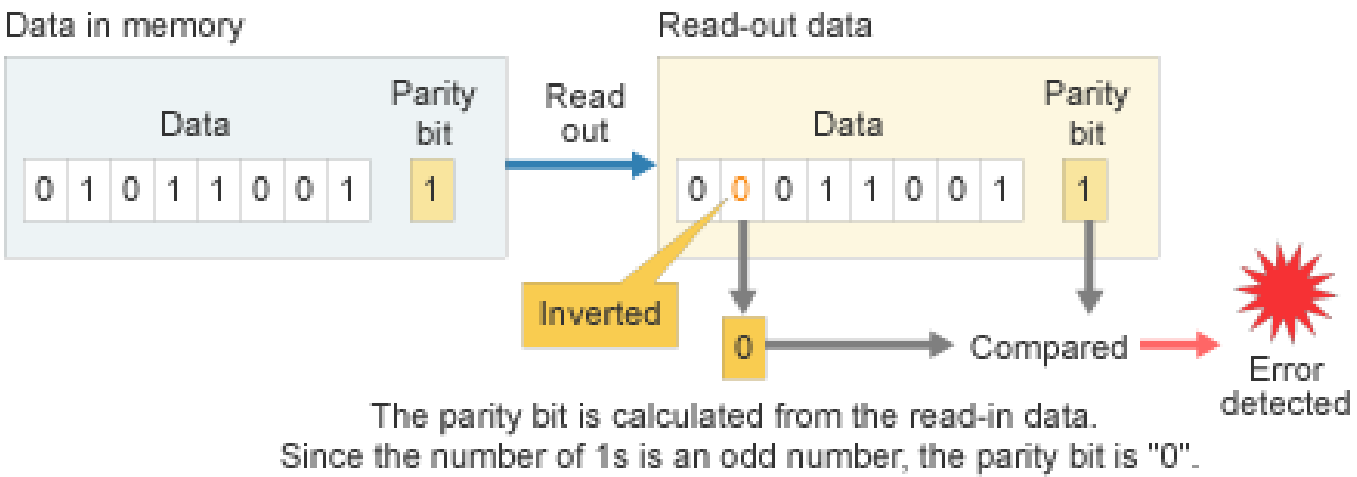
- i dati sono suddivisi in parole di N bit
- a ogni parola viene aggiunto un bit di controllo in modo tale che il numero $N + 1$ totale di bit uguali a 1 della codifica (pacchetto) risultante sia sempre pari o sempre dispari.

Il ricevitore:

- verifica la parità (pari o dispari)
- non può correggere eventuali errori
- non può rilevare errori che modificano un numero pari di bit in una codifica.

Codici di parità - Esempio

Original Data	Even Parity	Odd Parity
0 0 0 0 0 0 0 0	0	1
0 1 0 1 1 0 1 1	1	0
0 1 0 1 0 1 0 1	0	1
1 1 1 1 1 1 1 1	0	1
1 0 0 0 0 0 0 0	1	0
0 1 0 0 1 0 0 1	1	0



Questo esempio usa un codice di parità dispari.

Codice di correzione di Hamming

Il trasmettitore di un codice di Hamming:

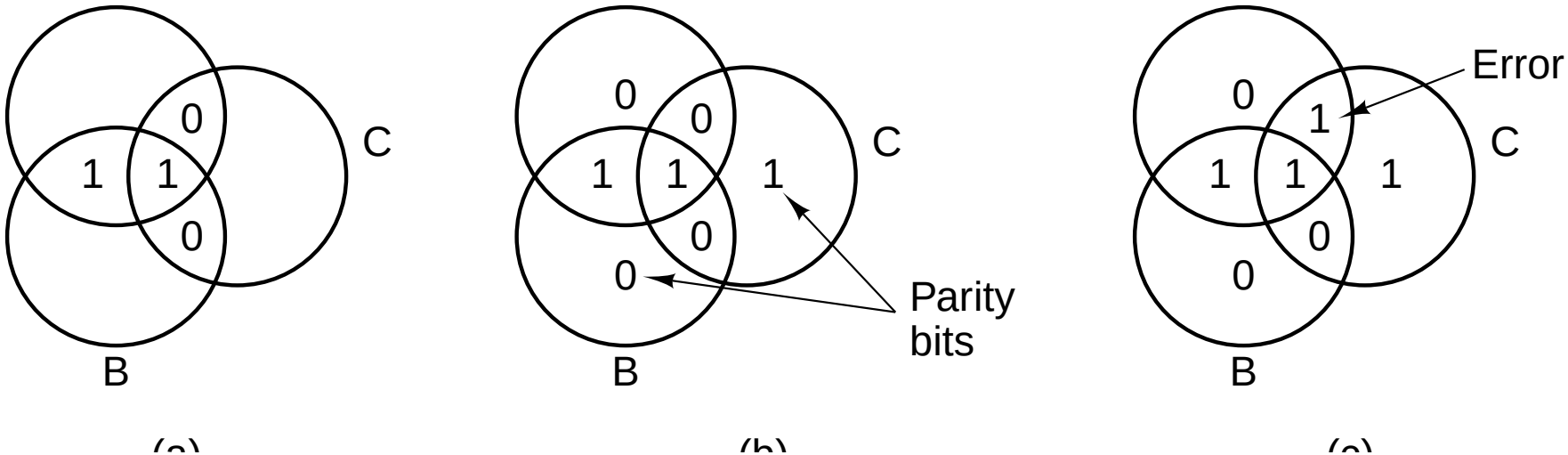
- divide la sequenza binaria di bit informativi in sottoinsiemi non disgiunti
- associa ogni bit ai sottoinsiemi a cui appartiene
- aggiunge un bit di parità per ogni sottoinsieme.

Il ricevitore:

- valuta la parità su ogni sottoinsieme
- se rileva uno o più errori elenca i sottoinsiemi contenenti bit errati
- se possibile, corregge l'errore negando il bit che appartiene a tutti e soli i sottoinsiemi errati.

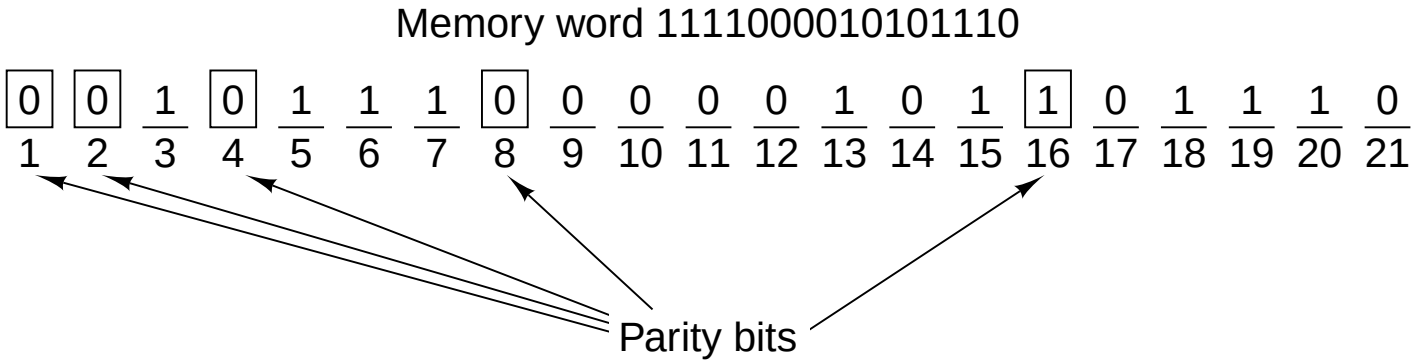
Es.: correzione di Hamming a 7 bit

Il codice di Hamming a 7 bit introduce 3 bit di controllo per ogni parola di 4 bit.



Codice di Hamming a N bit

É sempre $N = 2^k - 1$ con $K = 2, 3, \ldots$.



- etichetto in base 2 la posizione di ogni bit a partire dall'etichetta 1
- uso come bit di parità quelli la cui etichetta (posizione) è una potenza di 2: $1_2, 10_2, 100_2, \ldots$
- ogni bit di parità controlla i bit la cui etichetta contiene la cifra 1 nella stessa posizione
  Es. ($N = 7$): il bit in posizione 010 controlla i bit nelle posizioni 011, 110 e 111.

Costo di un codice

Introdurre informazione ridondante ha un costo: si utilizza più spazio (di canale oppure di memoria).

Costo di un codice:

$$\frac{\text{simboli ridondanti}}{\text{simboli utili}}.$$

Costo dei codici visti in precedenza:

- ripetizione doppia del carattere:

Costo di un codice

Introdurre informazione ridondante ha un costo: si utilizza più spazio (di canale oppure di memoria).

Costo di un codice:

$$\frac{\text{simboli ridondanti}}{\text{simboli utili}}.$$

Costo dei codici visti in precedenza:

- ripetizione doppia del carattere: costo 1 (100% simboli aggiuntivi)
- ripetizione tripla del carattere:

Costo di un codice

Introdurre informazione ridondante ha un costo: si utilizza più spazio (di canale oppure di memoria).

Costo di un codice:

$$\frac{\text{simboli ridondanti}}{\text{simboli utili}}.$$

Costo dei codici visti in precedenza:

- ripetizione doppia del carattere: costo 1 (100% simboli aggiuntivi)
- ripetizione tripla del carattere: costo 2 (200% simboli aggiuntivi)
- parità:

Costo di un codice

Introdurre informazione ridondante ha un costo: si utilizza più spazio (di canale oppure di memoria).

Costo di un codice:

$$\frac{\text{simboli ridondanti}}{\text{simboli utili}}.$$

Costo dei codici visti in precedenza:

- ripetizione doppia del carattere: costo 1 (100% simboli aggiuntivi)
- ripetizione tripla del carattere: costo 2 (200% simboli aggiuntivi)
- parità: $1/\text{dimensione parola}$
  Es. (pacchetti di 9 bit): $12,5\%$ simboli aggiuntivi.

Affidabilità di un codice

Nessun codice di correzione degli errori garantisce un'affidabilità assoluta:

- se una codifica ammissibile viene corrotta in una codifica ammissibile allora il ricevitore non rileva errori(!)
- nella pratica, se quasi tutti i bit trasmessi sono errati nessun codice di correzione funziona efficacemente.

Codice affidabile:

- improbabile che un errore non venga rilevato
- funziona anche con errori multipli.

Maggiore il numero di errori multipli gestibili, più affidabile è il codice.

Affidabilità - Esempi

Affidabilità dei codici visti in precedenza:

- ripetizione doppia del carattere:

Affidabilità - Esempi

Affidabilità dei codici visti in precedenza:

- ripetizione doppia del carattere: rileva 1 errore, non rileva 2 errori; non corregge un errore
- ripetizione tripla del carattere:

Affidabilità - Esempi

Affidabilità dei codici visti in precedenza:

- ripetizione doppia del carattere: rileva 1 errore, non rileva 2 errori; non corregge un errore
- ripetizione tripla del carattere: rileva 2 errori, corregge 1 errore; non rileva 3 errori, non corregge 2 errori
- parità:

Affidabilità - Esempi

Affidabilità dei codici visti in precedenza:

- ripetizione doppia del carattere: rileva 1 errore, non rileva 2 errori; non corregge un errore
- ripetizione tripla del carattere: rileva 2 errori, corregge 1 errore; non rileva 3 errori, non corregge 2 errori
- parità: rileva un numero dispari di errori sul pacchetto; non corregge un errore.

Es.: quanti errori rileva un codice di parità su un pacchetto di $N + 1$ bit?

Codifiche valide e non valide

Dato un codice binario a lunghezza fissa, distinguiamo tra

- codifiche valide: pacchetti di bit ottenibili da un dato iniziale applicando il codice
- codifiche non valide: tutti gli altri pacchetti.

In una comunicazione:

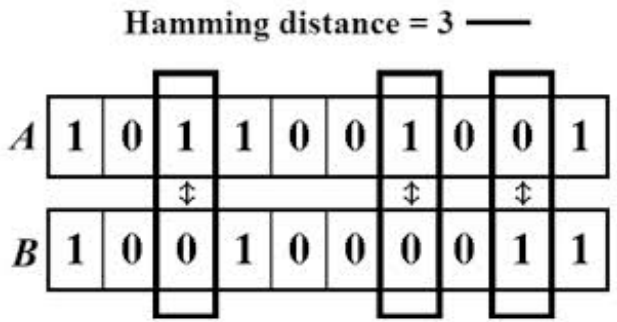
- il trasmettitore genera solo codifiche valide
- il ricevitore controlla se la codifica è valida. Una codifica non valida segnala un errore di trasmissione.

Distanza di Hamming

La distanza di Hamming fornisce una misura di "differenza" tra coppie di codifiche valide.

Nei codici binari (a lunghezza fissa)

Numero di coppie di bit allineati non coincidenti.



Più lontane due codifiche, maggiore il numero di errori necessari per trasformarne una nell'altra.

N.B.: non c'entra nulla col codice di Hamming!

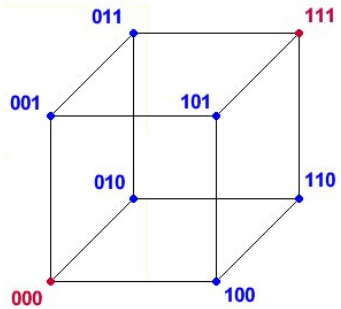
Distanza di Hamming di un codice

Distanza di Hamming $\mathcal{D}$ di un codice

Distanza minima tra tutte le coppie di codifiche valide distinte.

Esempi

- parità: $\mathcal{D} = 2$



- ripetizione tripla dei bit: $\mathcal{D} = 3$
- codice di Hamming a 7 bit: $\mathcal{D} = 3$
- codici Reed-Solomon (CD, DVD): $\mathcal{D} > 5$; correggono errori multipli.

Proprietà fondamentale della distanza di Hamming di un codice

Un codice con distanza di Hamming $\mathcal{D} = N$:

- rileva errori che modificano sino a $N - 1$ bit in una codifica

Proprietà fondamentale della distanza di Hamming di un codice

Un codice con distanza di Hamming $\mathcal{D} = N$:

- rileva errori che modificano sino a $N - 1$ bit in una codifica
- corregge errori che modificano sino a $(N - 1)/2$ bit in una codifica.

Locazioni e indirizzi di memoria

Una locazione di memoria occupa tipicamente 1 byte. Ogni locazione deve poter essere indirizzabile.

La dimensione usuale di un indirizzo è 32 o 64 bit.

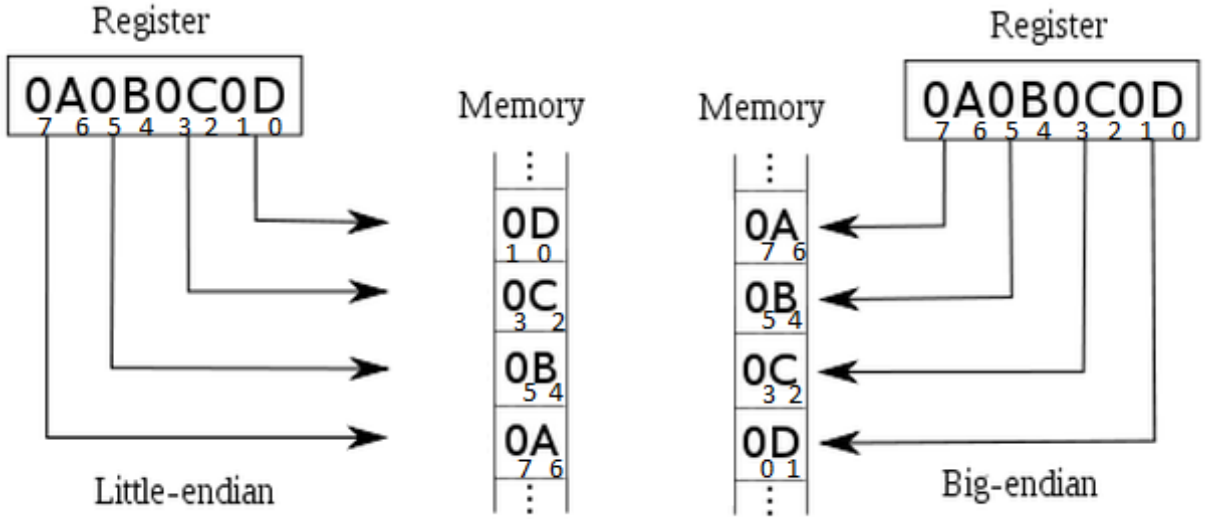
Come memorizzare un indirizzo? La soluzione più naturale è distribuirlo su 4 o 8 locazioni contigue.

Due modi possibili di memorizzare dati e indirizzi, a seconda del verso crescente della memoria

- little-endian: il dato/indirizzo è scritto in memoria partendo dal byte meno significativo
- big-endian: il dato/indirizzo è scritto in memoria partendo dal byte più significativo.

Convenzione little/big-endian

Processori Intel: little-endian. Altri: big-endian.



- Interi e reali devono essere riordinati quando passano da un processore little-endian a uno big-endian e viceversa.
- Le stringhe sono memorizzate allo stesso modo e quindi non devono essere riordinate quando passano da little- a big-endian e viceversa.