Codifica dei caratteri

Caratteri: informazioni contenuti in documenti testo

- cifre, lettere, simboli di punteggiatura
- simboli speciali: , #, $, %, &,), (
- caratteri speciali, informazioni di controllo: ritorno a capo, tabulazione, escape, controllo, ... Non visibili, definiscono la formattazione, contengono informazione non stampabili

Codici per caratteri

I caratteri vengono rappresentati mediante codici: si associa ad ogni carattere una sequenza di bit (un numero), Associazione arbitraria, alcune regole:

- cifre: codici consecutivi (trasformazione sequenza di cifre - numero),
- lettere: nel loro ordine (non necessariamente consecutive) (stabilire l'ordine lessicografico).

Molti codici:

- ASCII (standard 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 a larga diffusione (anni 60),
7 bit per carattere: 128 caratteri,

- 32 caratteri invisibili o di controllo, codici da 0 a 31, (carriage return, line-feed, back-space, cancel . . .) (escape, start of heading, end of transmission, . . .)
- 95 caratteri stampabili, codici da 32 a 127,

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

Problemi

- L’insieme di caratteri di controllo pensato per le telescriventi.
  Ritorno accapo = carriage return + line-feed.
  I sistemi operativi diversi gestiscono in diverso modo il ritorno accapo.
- Pochi i caratteri rappresentabili, non ci sono i caratteri non appartenenti all’inglese: lettere accentate, segni diacritici.

Estensioni ASCII

Unità d’informazione il byte (sequenze di 8 bit).
Naturale usare 8 bit per codificare un carattere.
Diverse estensioni del codice ASCII (standard), si conservano le prime 128 codifiche, si aggiungono nuovi 128 caratteri:

- ANSI (unix),
- MS DOS,
- MAC OS Roman.
- Lingue diverse usano caratteri diversi: devono usare estensioni diverse.

Standard 8859

Si mette ordine tra le diverse estensioni ASCII definendo un unico standard, pagina di codice

- IS 8859-1: ANSI, Latin 1, West Europe, IS 646;
- IS 8859-2: Latin 2, East Europe, lingue slave;
- IS 8859-3: Latin 3, South Europe, turco, esperanto . . . ;
- IS 8859-5: Cyrillic,
- IS 8859-6: Arabic,
- . . .

Problemi del ASCII esteso

- il software deve sapere su che pagina opera,
- non si possono mescolare le lingue,
- difficile gestire lingue con moltissimi caratteri: cinese e giapponese.

Problemi superabili con un codice un ampio insieme di caratteri codificabili.

UNICODE

Nato dall'accordo tra diverse aziende, (IS 10646)

- due (o più) byte per carattere, con 16 bit $2^{16} = 65.536$ caratteri,
- non solo caratteri ma anche simboli matematici, musicali, grafici.
- code point: il codice di un carattere,
- 65.5536 code point non sono sufficienti: nel mondo 200.000 simboli,
- la definizione non ancora completata.

UNICODE

- per semplificare la traduzione tra codifiche, più code point per lo stesso carattere,
- code point lasciati vuoti per future estensioni
- problemi culturali, UNICODE

Problemi UNICODE

- 65.536 caratteri non sono sufficienti:
  - ulteriore estensione: UCS (Universal Character Set),
  - sino a 21 bit per carattere, $\sim$ 2.000.000 di caratteri rappresentabili.
- documenti nella codifica UNICODE occupano il doppio dello spazio:
  - codifiche dell'UNICODE con lunghezze variabili: caratteri diversi usano un diverso numero di byte;
  - UTF-8 (UCS Transformation Format 8 bit) (UTF-7, UTF-16).

UTF-8 (UCS Transformation Format 8 bit)

Si possono rappresentare tutti i caratteri UCS e si generano file compatti:

- le codifiche UCS viene rappresentata con un numero variabile di byte,
- i caratteri ASCII standard (0-127) sono rappresentati con un byte (8 bit),
- gli altri caratteri: da 2 a 4 byte,
- compressione: caratteri standard (2^7) con 1 byte, caratteri particolari ($\sim 2^{11}$) con 2 byte, caratteri rari: 3-4 byte.
- UTF-16, usa 2 o 4 byte per carattere, migliore

UTF-8

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

Conclusioni

- Per quanto riguarda i caratteri la standardizzazione è limitata,
- è comune trovare documenti in diversi formati: Latin-1, UTF-8, UNICODE, MS-DOS, MAC-OS Roman.
- problemi nello scambio di dati.
- mail, documenti HTML devono specificare la codifica in cui sono scritti.

Codici di correzione degli errori

Nella trasmissione e memorizzazione dei dati si verificano degli errori:

- disturbi sulla linea (trasmissione),
- imperfezioni del supporto (memoria disco),
- radioattività (DRAM).

Necessità di meccanismi di protezione dagli errori, alcuni errori dell'hardware sono inevitabili.

Codici di correzione degli errori

Codici di correzione: meccanismi per controllare ed eventualmente correggere errori nei dati
Idea base: ridondanza di informazione.

- si aggiunge ai dati un informazione di controllo,
- si trasmette (o memorizza) più dati di quelli strettamente necessari,
- chi riceve, in base all'informazione ridondante, determina la presenza di errori.

Esempi dalla vita comune:

Nella comunicazione vocale non tutti i suoni vengono compresi correttamente.

Meccanismi di correzione per evitare errori:

- Sillabare: per comunicare un carattere si comunica un'intera parola,
  D $\Rightarrow$ Domodossola,
- Nel linguaggio parlato c'è più informazione di quella strettamente necessaria, il contesto, in cui appare una parola, spesso permette di correggere eventuali errori nella comprensione dei suoni.

Semplici esempi di codici

Trasmetto un testo ripetendo ogni carattere due volte:
`casa` $\Rightarrow$ `ccaassaa`
Se ricevo la sequenza: `ccaasraa` so che c'è un errore (ma non posso ricostruire il messaggio corretto).

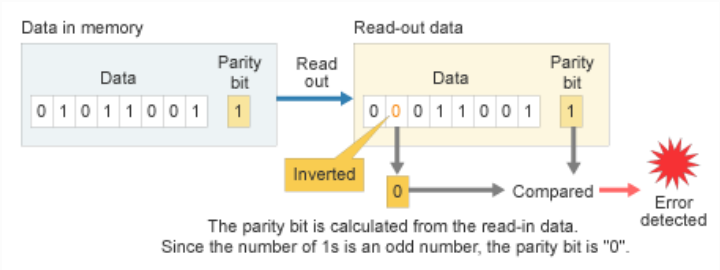
Per poter correggere eventuali errori: ripeto ogni carattere 3 volte
`cccaaasrsaaa` $\Rightarrow$ `casa`
`cccaaasrraaa` $\Rightarrow$ `cara`

Bit di parità

- I dati sono divisi in pacchetti (byte, parole)
- Ad ogni pacchetto viene aggiunto in bit di controllo,
  - in modo tale che il numero totale di bit 1 sia pari.
- Chi riceve il dato
  - può rivelare la presenza di errori (che modificano un solo bit in una parola),
  - non rivela errori multipli,
  - non può correggere eventuali errori.

Esempio

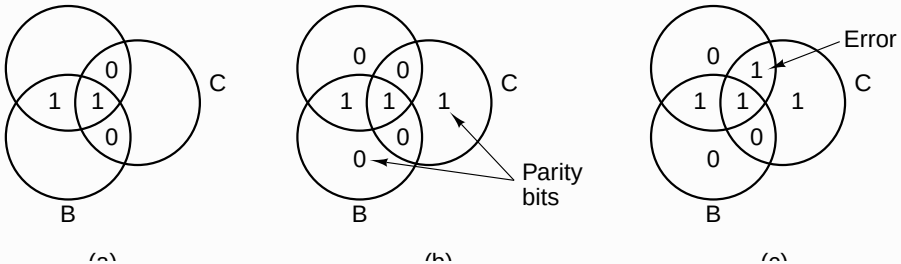
Original Data	Even Parity	Odd Parity
00000000	0	1
01011011	1	0
01010101	0	1
11111111	0	1
10000000	1	0
01001001	1	0



Questo esempio usa parità dispari.

Codice di correzione di Hamming

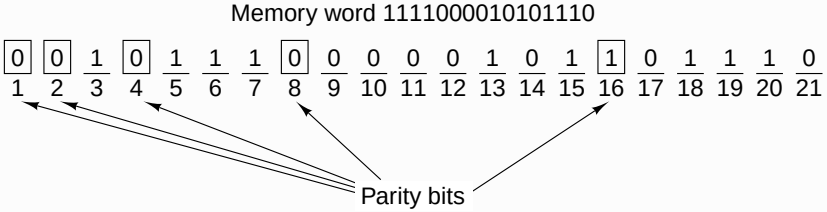
Codice di correzione per 4 bit.



- definiscono 3 sottoinsiemi di bit,
- un bit di parità per ogni sottoinsieme,
- ogni bit individuato dai sottoinsiemi a cui appartiene.

Codice di correzione di Hamming

Generalizzazione a *n* bit.



- enumero i bit con numeri nella notazione binaria, a partire da 1.
- uso come bit di parità quelli rappresentati da un sequenza con un solo 1, (le potenze di 2)
- ogni bit di parità controlla quei bit che contengono il corrispondete 1 nella loro

Costo di un codice di c. e.

Introdurre informazione ridondante costa: si utilizza più spazio disco, necessario maggior tempo per trasmettere.

Costo:

$$\frac{\text{dati ridondanti}}{\text{dati utili}}$$

Costo esempi precedenti:

- ripetizione doppia del carattere: costo 1
- ripetizione tripla del carattere: costo 2.
- bit di parità: $\frac{1}{\text{dim. pacchetto}}$.

Affidabilità di un codice di c. e.

Nessun codice di rilevazione (correzione) errore garantisce un'affidabilità assoluta:
se quasi tutti i bit trasmessi sono errati nessun codice funziona.

I codici permetto di rendere trascurabile la probabilità di un errore (non rivelato).

Codice affidabile: altamente improbabile che un errore non venga rilevato, funziona anche con errori multipli.
Più alto il numero di errori multipli gestibili, più affidabile è il codice.

Esempi precedenti

- ripetizione doppia del carattere: rivela 1 errore (2 errori sullo stesso carattere sfuggono);
- ripetizione tripla del carattere: rivela 2 errori (3 errori sfuggono), corregge 1 (due errori sono corretti in maniera sbagliata).
- bit di parità: rivela 1 errore.

Distanza di Hamming

Misura per l'affidabilità di un codice:
determina quanti errori (bit errati in un pacchetto di dati) il codice riesce a gestire,

Vengono usati due concetti:

- distanza tra parole,
- distinzione tra parole valide e parole non valide.

Distanza di Hamming tra parole

Una distanza misura quanto punti sono lontani

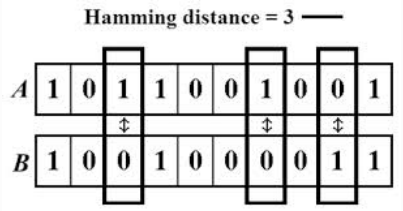
- nello spazio,
- nel tempo,
- in informatica (e in matematica), il concetto di *distanza* viene generalizzata ad altri ambiti,

Distanza di Hamming tra parole

La distanza di Hamming misura quanto due sequenze di simboli (parole), della stessa lunghezza, sono differenti tra loro.

Definizione

Il numero di simboli non coincidenti tra le due parole.



Alternativamente: il numero di errori necessari per trasformare una parole nell'altra.

Parole valide – non valide

In un codice di correzione errori distinguiamo tra:

- parole valide (lecite, legali): sequenze di bit ottenibili aggiungendo informazione di controllo ad un dato iniziale;
- parole non valide (illecite, non legali): tutte le altre sequenze di bit.

In una comunicazione:

- chi trasmette compone solo parole valide;
- chi riceve controlla se la parola è valida, una parola non valida segnala un errore di trasmissione.

Esempi

Nei codici visti in precedenza:

- ripetizione doppia del carattere; valide: “aa” , “bb”, “ss”; non valide “ab”, “as”, “cd”.
- ripetizione tripla del carattere; valide “aaa” , “bbb”, “sss”; non valide “aab”, “ssa”, “cde”.
- bit di parità; valide “01001011” , “00100001”, “11100111”; non valide “01001010” , “00100000”, “10100111”.

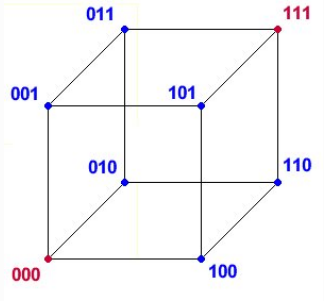
Distanza di Hamming di un codice

Definizione

Distanza di Hamming di un codice: distanza minima tra due diverse parole valide.

Distanza di Hamming di codici: esem.

- bit di parità ha distanza di Hamming 2;
- ripetizione tripla dei caratteri



  ha distanza di Hamming 3;
- codice Hamming (distanza 3);
- esistono codici con distanza di Hamming > 5

Distanza di Hamming

Un codice con distanza di Hamming n:

- scopre errori che modificano sino a $n-1$ bit in una parola;
- correggere errori che modificano sino a $(n-1)/2$ bit in una parola

Memorizzazione dati

Memoria: divisa in unità (locazioni) ogni indirizzo individua un unità

- dimensione standard di una locazione di memoria: 8 bit (1 byte),
- dimensione usuale per gli indirizzi: 32 bit

I calcolatori operano su parole di 32-64 bit (4-8 byte)

Come scrivere una parola in memoria?
Una parala viene distribuita su un gruppo di locazioni (da 1 byte) contigue.

Little-endian vs. Big-endian

Due modi possibili:

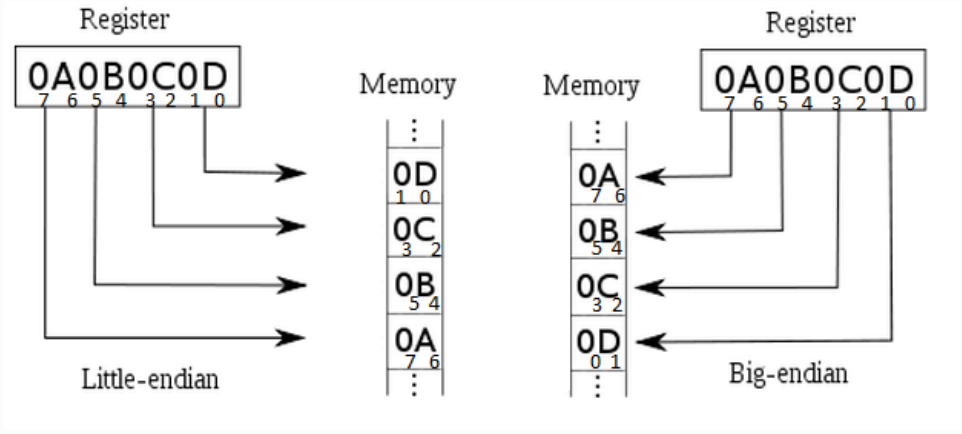
- big-endian: il primo byte della parola nella locazione con indirizzo più basso (la fine della parola viene scritta negli indirizzi più alti),
- little-endian: il primo byte della parola nella locazione con indirizzo più alto (la fine della parola viene scritta negli indirizzi più bassi).

Esempio

Inserire la parola “AB CD EF 01” nei byte con indirizzi da 4 a 7:

Big-endian:	Little-endian
4 ⇒ AB (10101011)	4 ⇒ 01
5 ⇒ CD (11001101)	5 ⇒ EF
6 ⇒ EF (11101111)	6 ⇒ CD
7 ⇒ 01 (00000001)	7 ⇒ AB

Esempio



Diverse scelte

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

Gli stessi dati vengono memorizzati in memoria in modo diverso.

Problemi nel trasferimento dati:

- alcuni dati, come numeri (interi reali), sono memorizzati in modo diverso,
  devono essere riordinati quando passano da un processore little-endian ad uno big-endian;
- altri dati, come stringhe (sequenze di caratteri), sono memorizzate allo stesso modo,
  non devono essere riordinati quando passano da little-endian a big-endian.