SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Altri codici importanti

Prof. Daniele Gorla

Il codice ASCII

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

ASCII è un acronimo per *American Standard Code for Information Interchange*

Nato nell'IBM nel 1961, diventa standard ISO (*International Organization for Standardization)* nel 1968.

Codifica con stringhe da 7 bit tutte le lettere maiuscole e minuscole dell'alfabeto inglese, le cifre decimali, i simboli di interpunzione, vari caratteri speciali…

- i 3 bit più significativi identificano il tipo (es.: 000 e 001 sono i caratteri speciali, 011 le cifre decimali, 100 e 101 le lettere maiuscole, etc.)
- i restanti 4 bit codificano il carattere in maniera monotona (se c'è un ordinamento naturale)

Es.: *a* viene prima di *d* nell'alfabeto → ASCII(*a*) < ASCII(*d*)
1 è minore di 5 →ASCII(1) < ASCII(5)

Tabella del codice ASCII a 7 bit

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

	Bits												
	b_7 b_6 b_5 →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	Column → / Row ↓	0	1	2	3	4	5	6	7
NULL	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
Start/end of Text	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
	0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
Horiz./Vert. Tab	1	0	0	1	9	HT	EM	)	9	I	Y	i	y
	1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
	1	0	1	1	11	VT	ESC	+	;	K	[	k	{
	1	1	0	0	12	FF	FC	,	<	L	\	l	|
Carriage return	1	1	0	1	13	CR	GS	-	=	M	]	m	}
	1	1	1	0	14	SO	RS	.	>	N	^	n	~
	1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Extended ASCII

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Problema del codice ASCII: 7 bit → 128 caratteri codificabili

Varie estensioni del codice a 8 bit (in questo modo ogni carattere era codificato con un byte)
→ Probl.: ogni compagnia ne adottava una propria
(IBM, Commodore, …), non necessariamente compatibili
con lo standard a 7 bit!!

Standard ISO (8859), composto da varie parti:

1. 256 caratteri per le lingue dell'Europa occidentale
2. 256 caratteri per le lingue dell'Europa centrale
3. 256 caratteri per le lingue dell'Europa meridionale
4. 256 caratteri per le lingue dell'Europa settentrionale
5. 256 caratteri per le lingue slave (cirillico)
6. 256 caratteri per l'arabo
7. 256 caratteri per il greco
8. 256 caratteri per l'ebraico
9. …

Il codice Unicode

Problema dello standard ISO 8859: stesso codice per caratteri diversi (di parti diverse).

1991: codice Unicode → codifica univoca di tutti i caratteri, di tutte le lingue vive e morte, ideogrammi, simboli matematici e chimici, Braille,…

Originariamente a 16 bit, oggi a 21 bit (ma con moltissime sequenze non usate).

Oggi supportato dalle principali piattaforme di programmazione e sistemi operativi (Java, XML, Corba,…).

Non è uno standard ma è continuamente aggiornato dall'Unicode Consortium.

Ammette versioni "semplificate" da 8 o 16 bit, contenenti solo i caratteri più frequentemente usati.

Codici rilevatori e correttori di errore

Indipendentemente da cosa rappresenta, una sequenza di bit in trasmissione su un mezzo fisico può venir alterat aniera imprevedibile:

Studieremo in breve alcune codifiche in grado di rilevare e, se possibile, correggere errori di trasmissione.

OSS.: se | {Parole di codice} | = | {messaggi da codificare} | , allora non è possibile rilevare (né tantomeno correggere) errori!
→ bisogna avere codifiche *ridondanti*
(in cui cioè |{parole di codice}| > |{messaggi da codificare}|)

N.B.: maggior ridondanza → maggiore protezione MA costo maggiore

Codice con bit di parità

Il codice rilevatore più semplice consiste nel codificare 2^n messaggi con n+1 bit → uso solo metà delle possibili parole di codice!

La codifica di una sequenza w di n bit è la sequenza (di n+1 bit) wb, dove:

$$b = \begin{cases} 0 & \text{se } w \text{ ha un numero pari di "1"} \\ 1 & \text{altrimenti} \end{cases}$$

Ogni codifica ha un numero pari di "1" → *codice a parità pari*
("spreco" metà parole di codice – quelle con un numero dispari di "1")

Rilevo 1 errore, correggo 0 errori.

Ha un numero dispari di "1"!! ERRORE!!! Ma dove??

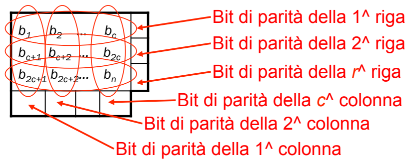
Ha un numero pari di "1"!! NON RILEVA GLI ERRORI!!!

Codice con parità longitudinale e trasversale (1)

Siano n i bit del messaggio e sia $n = r \times c$.

Rappresenta il messaggio come una matrice di r righe e c colonne, ognuna con un suo bit di parità → parole di codice lunghe n+r+c bit

Mess = $b_1 \ldots b_n$



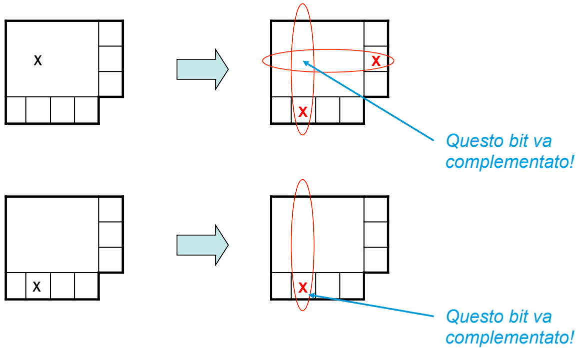
Quanti bit di ridondanza aggiungo?
il caso migliore si ha quando n è un quadrato perfetto → $r = c = \sqrt{n}$
il caso peggiore si ha quando n è un numero primo → $r = n$ e $c = 1$
Quindi, aggiungo un numero di bit di ridondanza (r+c) che varia tra $2\sqrt{n}$ e n+1.

Rilevo 2 errori, correggo 1 errore.

Codice con parità longitudinale e trasversale (2)

SAPIENZA Università di Roma Dipartimento di Informatica

Può correggere 1 errore:



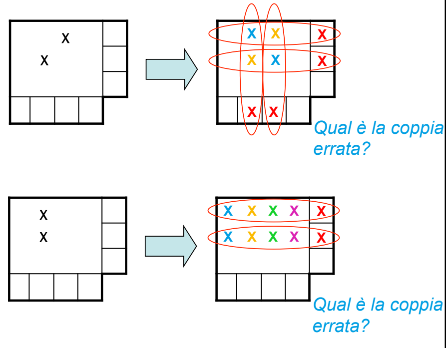
Codice con parità longitudinale e trasversale (3)

SAPIENZA Università di Roma Dipartimento di Informatica

Può rilevare 2 errori nel messaggio:

a) Non allineati

b) Allineati



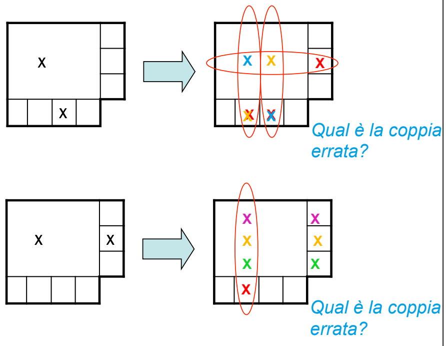
Codice con parità longitudinale e trasversale (4)

SAPIENZA Università di Roma Dipartimento di Informatica

Può rilevare 2 errori di cui uno nel messaggio e uno nei bit di parità:

a) Non allineati

b) Allineati

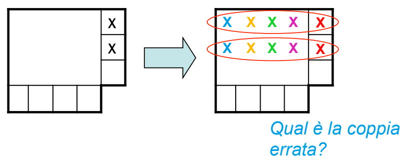


Codice con parità longitudinale e trasversale (5)

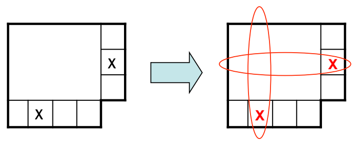
SAPIENZA Università di Roma Dipartimento di Informatica

Può rilevare 2 errori nei bit di parità:

a) Entrambi di riga/colonna



b) Uno di riga e uno di colonna



Se so che ci sono stati 2 errori, questo è l'unico caso in cui posso correggere; se so che ci sono stati AL PIU' 2 errori, non posso correggere.

Codice di Hamming

SAPIENZA Università di Roma Dipartimento di Informatica

Corregge 1 errore e ne rileva 2, ma con un numero inferiore di bit di controllo → usa sempre $\log_2 n + 1$ bit, invece che almeno $2\sqrt{n}$

Già per $n = 4$ è meglio ($\log_2 4 + 1 = 3$, $2\sqrt{n} = 4$)!

Vari codici, chiamati *codici di Hamming 2^n-a-$(n+1)$*: messaggi da 2^n bit e $n+1$ bit di controllo di parità.

Si può applicare a messaggi di lunghezza arbitraria:

- → se ho messaggi lunghi m, prendo il più piccolo n tale che $m \leq 2^n$, cioè prendo $n = \lceil \log_2 m \rceil$
- → metto $2^n - m$ "0" non significativi in testa ai messaggi

Codice di Hamming 4-a-3

SAPIENZA Università di Roma Dipartimento di Informatica

Idea: mischiare bit di controllo (nelle posizioni che sono potenze di 2) e bit di messaggio (nelle restanti posizioni):

Mess.: $m_1 m_2 m_3 m_4$

Mess.:			m_1		m_2	m_3	m_4
Posiz.:	1	2	3	4	5	6	7
Contr.:	c_1	c_2		c_3			

Controllo di parità su sottostringhe:

- c_1 controlla la parità di $m_1 m_2 m_4$;
- c_2 controlla la parità di $m_1 m_3 m_4$;
- c_3 controlla la parità di $m_2 m_3 m_4$.

	c_1	c_2	m_1	c_3	m_2	m_3	m_4
c_1	√		√		√		√
c_2		√	√			√	√
c_3				√	√	√	√

Imposto i bit di controllo in modo che ognuna di queste sottostringhe abbia parità pari (cioè un numero pari di "1")

Esempio

SAPIENZA Università di Roma Dipartimento di Informatica

Trovare la parola di codice di Hamming 4-a-3 per il messaggio 1011.

Mess.:			1		0	1	1
Posiz.:	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Contr.:	0	1		0			

Ha un numero pari di "1"

Ha un numero pari di "1"

Ha un numero dispari di "1"

Quindi la parola di codice associata al messaggio 1011 è 0110011.

Correggere 1 errore con il codice di Hamming

SAPIENZA Università di Roma Dipartimento di Informatica

Assumendo che ci sia stato al più un errore, possiamo identificarlo (e correggerlo) nel modo seguente:

Controlla la parità delle sottostringhe $c_1 m_1 m_2 m_4$, $c_2 m_1 m_3 m_4$ e $c_3 m_2 m_3 m_4$ (cioè, i caratteri in posizione 1-3-5-7, 2-3-6-7 e 4-5-6-7, rispettivamente):

$c_1 m_1 m_2 m_4$	$c_2 m_1 m_3 m_4$	$c_3 m_2 m_3 m_4$	
"1" dispari	"1" dispari	"1" dispari	**errore in m_4** $= \{c_1,m_1,m_2,m_4\} \cap \{c_2,m_1,m_3,m_4\} \cap \{c_3,m_2,m_3,m_4\}$
"1" dispari	"1" dispari	"1" pari	**errore in m_1** $= (\{c_1,m_1,m_2,m_4\} \cap \{c_2,m_1,m_3,m_4\}) \setminus \{c_3,m_2,m_3,m_4\}$
"1" dispari	"1" pari	"1" dispari	**errore in m_2** $= (\{c_1,m_1,m_2,m_4\} \cap \{c_3,m_2,m_3,m_4\}) \setminus \{c_2,m_1,m_3,m_4\}$
"1" pari	"1" dispari	"1" dispari	**errore in m_3** $= (\{c_2,m_1,m_3,m_4\} \cap \{c_3,m_2,m_3,m_4\}) \setminus \{c_1,m_1,m_2,m_4\}$
"1" dispari	"1" pari	"1" pari	**errore in c_1** $= \{c_1,m_1,m_2,m_4\} \setminus (\{c_2,m_1,m_3,m_4\} \cup \{c_3,m_2,m_3,m_4\})$
"1" pari	"1" dispari	"1" pari	**errore in c_2** $= \{c_2,m_1,m_3,m_4\} \setminus (\{c_1,m_1,m_2,m_4\} \cup \{c_3,m_2,m_3,m_4\})$
"1" pari	"1" pari	"1" dispari	**errore in c_3** $= \{c_3,m_2,m_3,m_4\} \setminus (\{c_1,m_1,m_2,m_4\} \cup \{c_2,m_1,m_3,m_4\})$
"1" pari	"1" pari	"1" pari	**nessun errore**

Esempio

SAPIENZA Università di Roma Dipartimento di Informatica

Stabilire se 0100011 è una la parola di codice di Hamming 4-a-3; in caso positivo, dire il messaggio associato; in caso negativo, identificare l'errore (assumendo che ce ne sia stato solo 1), correggerlo e restituire il messaggio.

Mess.:			0		0	1	1
Posiz.:	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Contr.:	0	1		0			

Ha un numero dispari di "1"

Ha un numero pari di "1"

Ha un numero dispari di "1"

- L'errore è in $c_1 m_1 m_2 m_4$;
- L'errore è in $c_2 m_1 m_3 m_4$;
- L'errore non è in $c_3 m_2 m_3 m_4$.

Quindi il bit errato è m_1

La parola di codice corretta è pertanto 0110011, da cui il messaggio associato è 1011.

Rilevare 2 errori con il codice di Hamming

SAPIENZA Università di Roma Dipartimento di Informatica

Assumendo che o ci sono stati 2 errori o nessuno, possiamo rilevare questi due situazioni, sempre controllando la parità delle sottostringhe formate dai caratteri in posizione 1-3-5-7, 2-3-6-7 e 4-5-6-7 ($c_1 m_1 m_2 m_4$, $c_2 m_1 m_3 m_4$ e $c_3 m_2 m_3 m_4$):

- se sono tutte corrette per parità, allora non c'è stato alcun errore;
- se almeno una di queste ha un errore di parità, ci sono stati 2 errori, ma non si riesce ad identificare la coppia di bit da correggere.

Es.: la stringa 0100111 non è una parola del codice di Hamming 4-a-3:
- 0100111 : numero pari di "1"
- 0100111 : numero dispari di "1"
- 0100111 : numero dispari di "1"

Con 1 errore, riesco a dire che la parola originale era 0100101.

Con 2 errori, non riesco a determinare univocamente la parola originale: potrebbe essere 0110011, 1100110 o anche 0001111.

Rilevare 3 errori con il codice di Hamming

SAPIENZA Università di Roma Dipartimento di Informatica

Se si hanno 3 errori, c'è la possibilità che una parola di codice si trasformi in un'altra parola di codice e quindi non si riesce a rilevare neanche il fatto che ci sono stati degli errori.

Es.: Se nella parola di codice 0110011 si corrompono i bit in posizione 3, 4 e 7 (cioè, se 0110011 diventa 0101010), si ottiene una stringa di bit che è ancora una parola del codice di Hamming 4-a-3! Infatti:
- **0**1**0**1**0**1**0** ha un numero pari di "1";
- 0**10**10**10** ha un numero pari di "1";
- 010**1010** ha un numero pari di "1".

Per rilevare/correggere più errori c'è bisogno di codici diversi e più sofisticati (non più basati sulla parità)