CODIFICA DELL'INFORMAZIONE E CODICI BINARI

Andrea Bobbio

Anno Accademico 2001-2002

La codifica dell'informazione

I sistemi di elaborazione operano al loro interno soltanto con segnali a due valori (binari).

I sistemi di elaborazione devono tuttavia scambiare informazioni con il mondo esterno in ingresso e in uscita.

Tali informazioni devono essere comprensibili ad un operatore umano, e assumono prevalentemente la forma di caratteri alfanumerici (numeri decimali, lettere dell'alfabeto, simboli di punteggiatura, simboli matematici etc..).

Tale apparente incomunicabilità fra i due linguaggi, viene ricomposta mediante l'adozione di opporture convenzioni (o *codici*), mediante le quali è possibile rappresentare in modo univoco un certo numero di simboli con configurazioni di bit prestabilite.

La codifica dell'informazione

Si dice *alfabeto* un insieme non vuoto e finito di simboli detti *caratteri*.

Ad esempio:

$\Rightarrow \mathcal{A} = \{0, 1\}$ è l'alfabeto binario.

$\Rightarrow \mathcal{B} = \{A, B, C, \ldots, X, Y, Z\}$ è l'alfabeto delle lettere latine maiuscole.

Una *stringa* o *parola* in un dato alfabeto è una successione di simboli (anche ripetuti) di quell'alfabeto.

Dato un insieme $\mathcal{I}$ di oggetti, si dice codifica dell'insieme $\mathcal{I}$ nell'alfabeto $\mathcal{L}$, un procedimento che permetta di stabilire una corrispondenza biunivoca fra gli elementi di $\mathcal{I}$ e un sottoinsieme di parole di $\mathcal{L}$.

La codifica dell'informazione

Siano dati due alfabeti qualunque, $\mathcal{A}$ di k simboli e $\mathcal{B}$ di m simboli:

$$\mathcal{A} = \{a_{k-1}, a_{k-2} \ldots, a_0\}$$

$$\mathcal{B} = \{b_{m-1}, b_{m-2} \ldots, b_0\}$$

è sempre possibile codificare l'insieme delle parole di $\mathcal{A}$ nell'alfabeto $\mathcal{B}$ e viceversa.

Procedimento:
Si possono interpretare i k simboli di $\mathcal{A}$ come un sistema di numerazione posizionale in base k, e quindi ogni parola di $\mathcal{A}$ può essere interpretata come un numero intero nel sistema posizionale di base k.

Analogamente, ogni parola di $\mathcal{B}$ può essere interpretata come un numero intero nel sistema posizionale di base m.

Codificare una parola di $\mathcal{A}$ in $\mathcal{B}$ equivale ad un cambiamento di base, da base k a base m.

Esempio di codifica

Sia $\mathcal{A} = \{A, B, C\}$ un alfabeto di 3 caratteri e $\mathcal{B} = \{Rosso, Verde, Giallo, Blu\}$ un alfabeto di 4 caratteri.

Si vuole codificare la seguente stringa di $\mathcal{A}$ in $\mathcal{B}$

$$B\,C\,C\,A\,B$$

Per interpretare i simboli di $\mathcal{A}$ come un sistema di numerazione posizionale in base 3, attribuisco ai simboli di $\mathcal{A}$ i seguenti valori:

$$A = 2 \quad ; \quad B = 1 \quad ; \quad C = 0$$

Per interpretare i simboli di $\mathcal{B}$ come un sistema di numerazione posizionale in base 4, attribuisco ai simboli di $\mathcal{B}$ i seguenti valori:

$$Rosso = 3 \;\; ; \;\; Verde = 2 \;\; ; \;\; Giallo = 1 \;\; ; \;\; Nero = 0$$

Da cui:

$$B\,C\,C\,A\,B = 1 \cdot 3^4 + 0 \cdot 3^3 + 0 \cdot 3^2 + 2 \cdot 3^1 + 1 \cdot 3^0 = 88_{(10)}$$

Vale la conversione $88_{(10)} \longrightarrow 1120_{(4)}$,
Per cui, infine:

$$(B\,C\,C\,A\,B)_{\mathcal{A}} \longrightarrow (Giallo\,Giallo\,Verde\,Nero)_{\mathcal{B}}$$

Lunghezza di un codice binario

Per rappresentare M simboli diversi *(parole)*, mutuamente esclusivi, con un codice binario, la lunghezza m della sequenza di bit del codice deve soddisfare la seguente relazione:

$$m \geq \lceil \log_2 M \rceil$$

Codici decimali pesati a 4 bit

Per rappresentare le 10 cifre distinte dei numeri decimali, occorrono codici binari a 4 bit.

Un codice decimale si dice *pesato* se la relazione che intercorre tra la cifra decimale D da rappresentare e l'insieme delle cifre binarie ad esso associato è del tipo:

$$D = \sum_{i=0}^{m-1} b_i \cdot p_i$$

dove:

$\rightarrow m$: numero di bit del codice

$\rightarrow b_i$: generico bit del codice

$\rightarrow p_i$: peso corrispondente.

Codice decimale BCD

Il codice decimale *BCD (Binary Coded Decimal)* è un codice pesato con pesi $(8-4-2-1)$.

Con questa codifica, ogni cifra decimale è rappresentata dal binario puro corrispondente, secondo la seguente tabella:

Decimale	*Codice BCD*			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Con questa codifica, le posizioni del codice da $(1\,0\,1\,0)$ a $(1\,1\,1\,1)$ non sono utilizzate.

Conversione decimale → binario BCD

Si converte ogni cifra decimale separatamente nel corrispondente codice *BCD* su 4 bit.

ESEMPIO:

Convertire il numero decimale $(5902)_{10}$ in codice binario *BCD*.

5	9	0	2
↓	↓	↓	↓
0 1 0 1	1 0 0 1	0 0 0 0	0 0 1 0

Conversione binario BCD → decimale

Si raggruppano le cifre binarie a 4 a 4 e si converte ciascun raggruppamento nella corrispondente cifra decimale secondo il codice *BCD*.

ESEMPIO:

Ricavare il valore decimale della sequenza di bit

0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 1

sapendo che corrisponde a una codifica *BCD*.

0 0 0 1	1 0 0 0	0 1 1 1	0 0 1 1
↓	↓	↓	↓
1	8	7	3

Codici a barre

I codici a barre, usati prevalentemente sui prodotti commerciali, sono particolari forme di codici binari.

Per informazione sui codici a barre è possibile consultare la pagina:

http://www.bizfonts.com/upc-ean/

Codici alfanumerici binari

Quando, oltre alle cifre decimali, si vogliono anche codificare caratteri (maiuscoli e minuscoli), punteggiatura, simboli matematici etc... occorre estendere il numero di bit m del codice.

Se M è il numero totale di caratteri per cui si vuole definire un codice binario, occorre prevedere una lunghezza di codice m data dalla relazione:

$$m \geq \lceil \log_2 M \rceil$$

Codice ASCII

Il codice ASCII è di gran lunga il codice alfanumerico più diffuso per lo scambio di informazioni fra sistemi di elaborazione.

La sigla ASCII significa American Standard Code for Information Interchange.

Il codice ASCII costituisce di fatto uno standard per la codifica dell'informazione nei sistemi di elaborazione.

Il codice ASCII standard è codificato su 7 bit, e quindi può rappresentare al massimo $2^7 = 128$ simboli diversi.

Esiste una versione del codice ASCII che usa un ottavo bit *(1 byte)*. Tale codice viene detto *Codice ASCII esteso*, ed è, ad esempio, usato nei personal computer IBM MS/DOS.

Codice ASCII

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char	Dec	Hx	Char	Dec	Hx	Oct	Char
0	0	000	NUL (null)	32	20	040	SPACE	64	40	100 @	96	60	140	‘
1	1	001	SOH (start heading)	33	21	041	!	65	41	101 A	97	61	141	a
2	2	002	STX (start of text)	34	22	042	"	66	42	102 B	98	62	142	b
3	3	003	ETX (end of text)	35	23	043	#	67	43	103 C	99	63	143	c
4	4	004	EOT (end of trans)	36	24	044	$	68	44	104 D	100	64	144	d
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105 E	101	65	145	e
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106 F	102	66	146	f
7	7	007	BEL (bell)	39	27	047	’	71	47	107 G	103	67	147	g
8	8	010	BS (backspace)	40	28	050	(	72	48	110 H	104	68	150	h
9	9	011	TAB (horizontal tab)	41	29	051	)	73	49	111 I	105	69	151	i
10	A	012	LF (NL line feed)	42	2A	052	*	74	4A	112 J	106	6A	152	j
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113 K	107	6B	153	k
12	C	014	FF (NP form feed)	44	2C	054	,	76	4C	114 L	108	6C	154	l
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115 M	109	6D	155	m
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116 N	110	6E	156	n
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117 O	111	6F	157	o
16	10	020	DLE (data link esc)	48	30	060	0	80	50	120 P	112	70	160	p
17	11	021	DC1 (device contr 1)	49	31	061	1	81	51	121 Q	113	71	161	q
18	12	022	DC2 (device contr 2)	50	32	062	2	82	52	122 R	114	72	162	r
19	13	023	DC3 (device contr 3)	51	33	063	3	83	53	123 S	115	73	163	s
20	14	024	DC4 (device contr 4)	52	34	064	4	84	54	124 T	116	74	164	t
21	15	025	NAK (negative ack)	53	35	065	5	85	55	125 U	117	75	165	u
22	16	026	SYN (synchr idle)	54	36	066	6	86	56	126 V	118	76	166	v
23	17	027	ETB (end trans block)	55	37	067	7	87	57	127 W	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130 X	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131 Y	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132 Z	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133 [	123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134 \	124	7C	174	|
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135]	125	7D	175	}
30	1E	036	RS (rec separator)	62	3E	076	>	94	5E	136 ^	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137 _	127	7F	177	DEL

Codice UNICODE

Lo standard Unicode è stato introdotto per rappresentare caratteri di testo in sistemi informatici, ed è stato assunto come standard internazionale con la sigla ISO/IEC 10646.

Lo standard è reperibile al sito:

http://www.unicode.org/

Unicode propone uno standard per rappresentare i caratteri e simboli di tutti i linguaggi scritti, simboli matematici etc ....

Unicode usa un codice a 16 bit, con cui è possibile codificare $2^{16} = \sim 65,000$ caratteri distinti.

Codici Ridondanti

Per codificare M simboli distinti *(parole)* con un codice binario, occorrono $m \geq \lceil \log_2 M \rceil$ bit.

Un codice si dice ridondante, quando codifica gli M simboli distinti con $n = m+r$ bit, cioè usando r bit aggiuntivi rispetto agli m bit strettamente richiesti dalla codifica binaria.

L'aggiunta di bit di ridondanza permette di costruire codici che consentono di controllare eventuali errori di trasmissione.

Si hanno due tipi di codici ridondanti:

- ◇ Codici a rivelazione di errore - Consentono di individuare la presenza di un errore;
- ◇ Codici a correzione di errore - Consentono non solo di individuare la presenza di un errore, ma anche di identifcarne la posizione in modo da poterlo correggere.

Codice BCD con bit di parità

Per costruire un codice *BCD* a rivelazione di errore, si può aggiungere ad ogni parola codice un bit ridondante ($r = 1$) detto di *parità*.

Il bit di *parità* viene posto a 0 o a 1, in modo tale che la somma degli uni nella parola codice sia pari:

Decimale	*Codice BCD*				*Bit parità*
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0

Conversione decimale → binario BCD con bit di parità

Si converte ogni cifra decimale separatamente nel corrispondente codice *BCD* e si aggiunge il quinto bit di parità come da tabella precedente.

ESEMPIO:

Convertire il numero decimale $(4765)_{10}$ in codice binario *BCD* con bit di parità.

4	7	6	5
↓	↓	↓	↓
0 1 0 0 1	0 1 1 1 1	0 1 1 0 0	0 1 0 1 0

Codici Ridondanti - Distanza di Hamming

La *distanza di Hamming* fra due parole codice, si ottiene contando il numero di bit diversi in posizioni corrispondenti:

$$\underline{1}\ 0\ 1\ \underline{0}\ \underline{0}\ 1$$

Distanza di Hamming=3

$$\underline{0}\ 0\ 1\ \underline{1}\ \underline{1}\ 1$$

In un codice a rivelazione di errore la distanza di Hamming fra due parole codice deve essere ≥ 2, in quanto un errore singolo deve produrre una sequenza di bit che non appartiene a nessuna parola codice.

In un codice a correzione di errore singolo la distanza di Hamming fra due parole codice deve essere ≥ 3, in quanto un errore singolo deve produrre una sequenza di bit che ha distanza di Hamming 1 dalla parola originaria e distanza di Hamming almeno 2 rispetto a ogni altra parola codice, in modo da identificare univocamente la parola originaria.

Codici di Hamming

Il *codice di Hamming* fornisce una procedura sistematica per generare codici ridondanti correttivi, tali che sia palese l'indicazione degli eventuali bit errati nella parola codice.

Verrà considerato solo il caso di *codice di Hamming* autocorrettivo per bit singolo (in grado cioè di correggere un eventuale errore su un solo bit).

Siano:	m	bit di parola
	r	bit di ridondanza
	$n = m + r$	bit di parola codice

Ognuna dell 2^m parole legali, ha n parole codice errate a distanza di Hamming 1, ottenute cambiando un bit alla volta nella parola originaria.

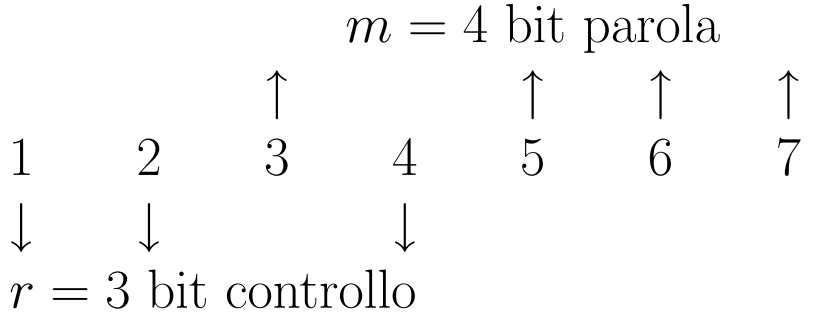
Ognuna dell 2^m parole legali richiede $(n + 1)$ configurazioni di bit dedicate. Per cui deve essere:

$$\begin{aligned} 2^n &\geq 2^m (n + 1) \\ 2^m \cdot 2^r &\geq 2^m (m + r + 1) \\ 2^r &\geq (m + r + 1) \end{aligned}$$

Codice di Hamming Autocorrettivo (4+3)

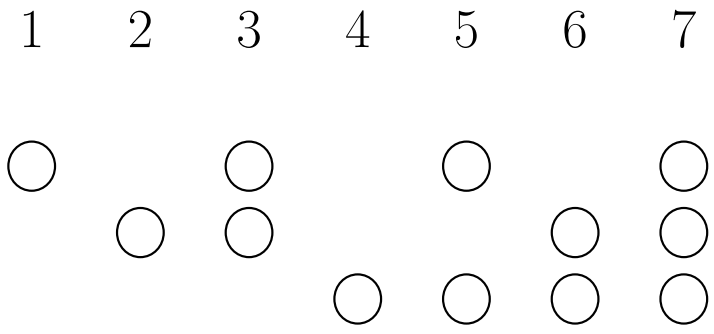
Con $m = 4$ si devono avere $r = 3$ bit di ridondanza.

Si dispongono gli m bit della parola e gli r bit di ridondanza (o di controllo) nel seguente modo:



Si dispongono i bit di controllo nelle posizioni corrispondenti a potenze di 2 (1, 2, 4, 8, ...) a partire dalla più significativa.

Ad ogni bit di controllo viene assegnato un valore di parità sulle sequenze di bit individuate come dalla seguente tabella:



Codice di Hamming Autocorrettivo (4+3)

Ogni bit di controllo deve garantire la parità sulle sequenze di bit della tabella:

r_1	r_2	m_1	r_3	m_2	m_3	m_4
1	2	3	4	5	6	7
○		○		○		○
	○	○			○	○
			○	○	○	○

bit 1 $\rightarrow$ controlla la parità sui bit 1, 3, 5 e 7.

bit 2 $\rightarrow$ controlla la parità sui bit 2, 3, 6 e 7.

bit 4 $\rightarrow$ controlla la parità sui bit 4, 5, 6 e 7.

Il generico bit b è controllato dai bit r_i secondo la seguente tavola della verità:

r_3	r_2	r_1	b
		X	1
	X		2
	X	X	3
X			4
X		X	5
X	X		6
X	X	X	7

Codice di Hamming Autocorrettivo (4+3)

La rilevazione e correzione di un eventuale bit errato avviene controllando il valore di parità dei bit di controllo.

Se il valore di parità del bit di controllo è corretto si pone a zero il valore nella tabella precedente; se non è corretto si pone a 1 il valore.

ESEMPIO:

Se si riscontra errata la parità nei bit r_1 e r_3, il bit errato è $b = 5$. Infatti:

r_3	r_2	r_1	b
1	0	1	5

Se si riscontra errata la parità nei bit r_1 e r_2, il bit errato è $b = 3$. Infatti:

r_3	r_2	r_1	b
0	1	1	3

Codice di Hamming Autocorrettivo (4+3)

ESEMPIO: Si determini il codice di Hamming per la parola su $m = 4$ bit:

$$1\ 0\ 1\ 1$$

Si mettano i bit assegnati nelle posizioni corrette:

r_1	r_2	m_1	r_3	m_2	m_3	m_4
1	2	3	4	5	6	7
		1		0	1	1

Deve essere:

- $r_1 = 0$ perchè sia pari la sequenza 1, 3, 5, 7.
- $r_2 = 1$ perchè sia pari la sequenza 2, 3, 6, 7.
- $r_3 = 0$ perchè sia pari la sequenza 4, 5, 6, 7.

Il codice di Hamming completo per la parola data risulta:

r_1	r_2	m_1	r_3	m_2	m_3	m_4
1	2	3	4	5	6	7
0	1	1	0	0	1	1

Codice di Hamming Autocorrettivo (4+3)

ESEMPIO: Si supponga di avere la seguente parola codice, e di voler controllare se è corretta:

r_1	r_2	m_1	r_3	m_2	m_3	m_4
1	2	3	4	5	6	7
0	1	0	0	0	1	1

Si verifica la correttezza di parità dei bit di controllo, e si individua univocamente l'eventuale bit errato:

- La sequenza 1, 3, 5, 7 è dispari (r_1 errato).
- La sequenza 2, 3, 6, 7 è dispari (r_2 errato).
- La sequenza 4, 5, 6, 7 è pari (r_3 corretto).

La parola codice data contiene un bit errato che è il bit $b = 3$. Infatti:

r_3	r_2	r_1	b
0	1	1	3

La parola codice corretta risulta quindi essere:

1	2	3	4	5	6	7
0	1	1	0	0	1	1