CAPITOLO I
SISTEMI DI NUMERAZIONE E CODICI

1.1) Sistema di numerazione decimale.

E' detto sistema di numerazione l'insieme di un numero finito di simboli e delle regole che assegnano uno e un solo valore numerico ad ogni stringa formata con i simboli stessi.

Tutti i moderni sistemi di numerazione sono posizionali; i simboli (detti anche cifre) vengono ordinati secondo valori via via crescenti di un'unita', ma il numero rappresentato da una stringa di tali simboli dipende anche dalla loro posizione reciproca. Le cifre del sistema numerico decimale sono i ben conosciuti simboli 0,1,...,9 e il significato di ciascuna stringa

$$A_n A_{n-1} A_{n-2} \ldots\ldots\ldots\ldots A_o$$

e' il valore numerico

$$N = A_n.10^n + A_{n-1}.10^{n-1} + A_{n-2}.10^{n-2} + \ldots\ldots + A_o.10^0$$

Usualmente per indicare numeri con una parte frazionaria, inferiore all'unita', si posiziona nella stringa una virgola per separare i simboli relativi a potenze di 10 di esponente maggiore o uguale a zero da quelli relativi a potenze di 10 di esponente minore di zero.

1.2) Sistemi di numerazione a base qualsiasi.

Nel sistema di numerazione posizionale decimale il numero 10 prende il nome di base numerica. Tuttavia quanto detto per il sistema decimale e' facilmente estendibile a sistemi di numerazione posizionale in cui la base non sia 10, ma un numero B qualsiasi.

In tal caso le cifre sono rappresentate da B simboli diversi e vanno da 0 a B-1. Indicando con a_k la generica cifra in k-esima posizione in una stringa, il significato della stringa stessa e:

$$N = A_n.B^n + A_{n-1}.B^{n-1} + \ldots\ldots + A_o.B^o$$

Anche il significato della virgola rimane immutato. Le basi numeriche piu' comuni, oltre B=10, sono B=2, B=8, B=16. La base 2 e' la piu' piccola teoricamente possibile per un sistema di numerazione posizionale. Le sue cifre sono rappresentate con i simboli 0 e 1; ciascuna cifra appartenente ad una stringa prende il nome di bit (binary digit). Il sistema a base 2 e' in sostanza il solo usato nei calcolatori numerici. Infatti, a parte altre considerazioni che comunque ne consiglierebbero l'uso, esistono numerosi dispositivi elettronici in grado di rappresentare mediante due stati di funzionamento le due cifre binarie.

I sistemi di numerazione a base 8 e 16 hanno come cifre rispettivamente i simboli 0,1,2,3,4,5,6,7 e 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. La loro diffusione e' giustificata dal fatto che la conversione tra essi e il sistema binario e' particolarmente semplice e che permettono di esprimere un valore numerico in maniera molto piu' compatta che non il sistema binario.

Onde evitare confusioni, ogni volta che si usano sistemi di numerazione diversi, e' conveniente indicare con opportuni pedici la base numerica. Ad esempio:

$$110111_2 = 55_{10}$$

1.3) Conversione tra sistemi a base diversa.

a) CONVERSIONE DI NUMERI INTERI.

Il metodo di conversione piu' semplice e' quello che fa ricorso alla successiva divisione per la base numerica. Si abbia infatti il numero

$$N = A_n.B^n + A_{n-1}.B^{n-1} + ... + A_o.B^o$$

Ad ogni successiva divisione si ottiene un quoziente e un resto e le divisioni vanno iterate fino ad ottenere un quoziente nullo. La stringa dei resti, letta in ordine inverso a quello in cui e' stata ottenuta, rappresenta la numerazione posizionale del numero N nella base B. Infatti dalle divisioni successive si ottiene:

Quoziente	**Divisore**	**Resto**
$N = A_n.B^{n-1} + A_{n-1}.B^{n-2} + ... + A_1.B^o$	B	A_o
$N = A_n.B^{n-2} + A_{n-1}.B^{n-3} + ... + A_2.B^o$	B	A_1
…….	…..	…..
$A_n.B^o$	B	A_{n-1}
0	B	A_n

La notazione posizionale e' quindi:

$$A_n A_{n-1} A_{n-2} .....A_o$$

Da quanto detto risulta evidente che la conversione tra una numerazione posizionale in base 2 e quelle in base 8 o 16, che sono a loro volta potenze di 2, e' particolarmente semplice. E' sufficiente infatti raggruppare i bit, a partire dalla cifra meno significativa, in gruppi di tre o quattro cifre e convertire ciascun gruppo nella corrispondente cifra ottale o esadecimale.

Ad esempio si voglia convertire in base 2, 8 e 16 il numero decimale 157. Secondo la procedura delle divisioni successive si ottengono i risultati illustrati nella tabella che segue.

Base 2		**Base 8**		**Base 16**	
Quoziente	**Resto**	**Quoziente**	**Resto**	**Quoziente**	**Resto**
157	1	157	5	157	D
78	0	19	3	9	9
39	1	2	2	0	
19	1	0			
8	1				
4	0				
2	0				
1	1				
0					

$157_{10} = 10011101_2$ $\qquad$ $157_{10} = 235_8$ $\qquad$ $157_{10} = 9D_{16}$

Come detto, la conversione a base 8 e a base 16 si puo' ottenere anche raggruppando opportunamente le cifre binarie.

Anche la conversione inversa e' immediata e si ottiene utilizzando la stessa tecnica.

b) CONVERSIONE DI FRAZIONI.

La conversione avviene per successive moltiplicazioni per la base numerica. I valori degli interi ottenuti costituiscono, nell'ordine, le cifre del numero nella base voluta; i soli valori frazionari vengono usati nelle successive moltiplicazioni. Infatti:

Parte intera	**Frazione**	**Moltiplicatore**
---	$A_{-1}.B^{-1} + ... + A_{-n}.B^{-n}$	B
A_{-1}	$A_{-2}.B^{-1} + ... + A_{-n}.B^{-n+1}$	B
---	----------	---
A_{-n}	----------	---

Quale esempio si voglia convertire in binario il numero decimale 0,6375

Frazione	**Parte intera**
0,6375 . 2 = 1,275	1
0,275 . 2 = 0,550	0
0,550 . 2 = 1,100	1
0,100. 2 = 0,200	0
0,200 . 2 = 0,400	0
0,400 . 2 = 0,800	0
0.800 . 2 = 1,600	1
0.600 . 2 = 1,200	1

Limitandosi quindi a tale numero di cifre significative si ha:

$$0,6375_{10} = 0,10100011_2$$

1.4) Metodi di conversione da binario a decimale.

I metodi di conversione esposti al paragrafo precedente sono di applicazione particolarmente semplice solo nel caso in cui si debba convertire un numero decimale in un'altra base numerica. E' infatti molto facile eseguire le operazioni descritte per la notevole familiarita' che si ha di solito con l'aritmetica decimale. Cio' non e' altrettanto vero nel caso di basi numeriche diverse.

Particolarmente utile e' allora illustrare alcuni metodi di conversione da sistema di numerazione binario a decimale.

Il metodo piu' ovvio e' quello che prevede l'espansione del numero in potenze di 2, applicabile a qualunque numero binario.

$$11010{,}011_2 = 1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 + 0.2^{-1} + 1.2^{-2} + 1.2^{-3} = 26{,}375_{10}$$

Un metodo piu' rapido, applicabile ai numeri interi e' il seguente: si raddoppia il bit piu' significativo e vi si somma quello immediatamente seguente; si raddoppia tale somma e vi si aggiunge il terzo bit e cosi' via fino ad esaurimento dell'intera stringa.

1 1 0 1 1 0

1 . 2 + 1

3 . 2 + 0

6 . 2 + 1

13 . 2 + 1

27 . 2 + 0

54

1.5) Aritmetica binaria.

1.5.1) Addizione binaria.

Le regole dell'addizione di due cifre binarie sono le seguenti:

$0 + 0 = 0$	
$0 + 1 = 1$	
$1 + 0 = 1$	
$1 + 1 = 10_2$	cioe' 0 con riporto di un'unita' al rango immediatamente superiore.

1.5.2) Sottrazione binaria.

Le regole della sottrazione di due cifre binarie sono:

$0 - 0 = 0$	
$1 - 0 = 1$	
$1 - 1 = 0$	
$0 - 1 = 1$	con riporto negativo di un'unita' al rango immediatamente superiore.

La sottrazione binaria, come del resto la sottrazione in qualsiasi sistema di rappresentazione numerico, puo' essere eseguita in modo diverso da quello appena illustrato qualora i numeri negativi vengano rappresentati in notazione complementata.

1.5.3) Complemento a B e a B-1.

In generale il complemento a B di un numero N di n cifre e' dato da:

$$C_B = B^n - N$$

Si intende invece complemento a B - 1 la quantita'

$$C_{B-1} = B^n - N - 1$$

Ne risulta allora che:

$$C_B = C_{B-1} + 1$$

Questa semplice proprieta' permette di ottenere il complemento a 2 di un numero binario molto facilmente. E' sufficiente infatti negare ogni bit del numero di partenza e sommare al risultato 1. Si voglia da esempio complementare a 2 il numero 1 1 0 1. Si ha:

1 1 0 1	13_{10}
0 0 1 0	dopo la negazione
0 0 0 1	somma di un'unita'
0 0 1 1	$3_{10} = 2^4 - 13_{10}$

Se come rappresentazione dei numeri negativi si adotta la rappresentazione complementata, l'operazione di sottrazione puo' essere eseguita nel modo che segue. Sia $N_1 - N_2$ la sottrazione da calcolare e siano sia N_1 che N_2 maggiori di zero. Si sostituisca a $- N_2$ la sua rappresentazione complementata $B^n - N_2$ e si esegua la somma tra N_1 e tale complemento. Si ottiene:

$$R = N_1 + B^n - N_2 = B^n + (N_1 - N_2)$$

Si possono verificare due casi: quello in cui $N_1 - N_2$ sia maggiore o uguale a zero e quello in cui $N_1 - N_2$ sia minore di zero. Si ricordi inoltre che ambedue i numeri sono minori di B^n .

Nel primo caso si avra' $R \geq B^n$ e quindi il rango relativo a B^n sara' riempito da una cifra diversa dallo zero, mentre gli altri ranghi, meno significativi, conterranno in notazione posizionale il valore $N_1 - N2$.

E' sufficiente pertanto ignorare il rango della potenza n-esima di B per ottenere il risultato voluto.

Nel secondo caso si ha $R < B^n$ e quindi il rango relativo a B^n sara' riempito da uno zero, mentre negli altri ranghi, meno significativi, sara' contenuto il valore

$$B^n + (N_1 - N_2) = B^n - (N_2 - N_1)$$

che e' proprio il complemento a B del numero $(N_2 - N_1) \geq 0$ assunto come rappresentativo del numero negativo $-(N_2 - N_1)$ risultato dell'operazione di sottrazione.

Si noti che nella rappresentazione dei numeri negativi mediante complemento a B lo zero si intende appartenente al corpo dei numeri positivi e non e' possibile complementarlo, mentre l'insieme dei numeri negativi comprende un intero in piu' di quello dei numeri positivi rappresentabili con gli stessi bit[1]. Ad esempio il massimo numero positivo rappresentabile con quattro bit e', considerando il quarto bit quello di segno:

$$0\ 1\ 1\ 1 = 7_{10}$$

mentre il piu' piccolo numero negativo nella rappresentazione complemento a 2 e':

$$1\ 0\ 0\ 0 = -\ 8_{10}$$

Esso non e' tuttavia complementabile per ottenerne il valore assoluto in quanto non e' rappresentabile nel campo dei numeri positivi con il numero di bit a disposizione.

Il risultato di un'operazione di sottrazione realizzata tramite il complemento e' esatto se non si e' avuto alcun riporto ne' nel bit di segno che al di fuori della parola di memoria oppure se si e' verificato in entrambi; e' errato se si e' avuto un riporto solo. A titolo di esempio si consideri una parola di memoria di 5 bit. Il massimo numero positivo rappresentabile e' quindi 15.

0 0 1 1 1 +	7 +	
0 0 1 0 1	5	
0 1 1 0 0	12	nessun riporto

[1] *E' necessario a questo punto far notare che nei calcolatori elettronici, che pur si avvalgono del complemento per eseguire la sottrazione, le cose vanno in maniera lievemente diversa.*
Infatti:

1.1) Tutti i numeri hanno la stessa lunghezza di stringa, essendo questa fissata dalla lunghezza in bit della parola di memoria.

1.2) Il bit piu' significativo rappresenta il valore rispetto al quale si complementa e pertanto assume il significato di bit di segno, avendo valore nullo per i numeri positivi e valore 1 per i numeri negativi, rappresentati nella forma complemento a 2.

1.3)Il massimo numero rappresentabile su n posizioni non e' dunque 2^n -1, ma 2^{n-1} -1, ed il risultato di un'operazione e' significativo se e solo se e' rappresentabile entro tale limite.

0 1 1 1 1 +	15 +	
0 1 1 1 1	15	
1 1 1 1 0	30	un riporto solo. Il risultato verrebbe interpretato come –2
1 0 0 1 1 +	- 13 +	
1 0 0 0 1	- 15	
1 0 0 1 0 0	- 28	un riporto solo. Il risultato verrebbe interpretato come 4
1 1 1 0 0 +	4 +	
1 1 0 0 0	- 8	
1 1 0 1 0 0	- 12	doppio riporto. Il risultato viene correttamente interpretato come - 12.

1.5.4) Moltiplicazione binaria.

Le regole della moltiplicazione binaria sono:

0 . 0 = 0
0 . 1 = 0
1 . 0 = 0
1 . 1 = 1

Nel caso di moltiplicazione di numeri di piu' cifre il modo di procedere e' del tutto simile a quello usato per la numerazione decimale.

1.5.5) Divisione binaria.

Ha un interesse puramente teorico in quanto vengono eseguite di solito per sottrazioni successive. Valgono comunque le medesime regole della divisione decimale.

1.6) I codici.

Si definisce codice un insieme **[**C] di parole adottato per rappresentare gli elementi di un insieme **[C*]** . E' ad esempio un codice l'insieme delle parole di una lingua; simboli di questo codice sono le lettere dell'alfabeto, mentre parole di codice sono le combinazioni di lettere che hanno significato.

Quale codificazione si intende l'operazione con la quale ad una parola del codice **[**C] viene fatto corrispondere un elemento dell'insieme **[C*]**.

Un codice e' non ambiguo se la corrispondenza tra le sue parole e gli elementi di **[C*]** e' univoca; ambiguo se almeno una parola di **[**C] rappresenta due o piu' elementi di **[C*]**.

Affinche' un codice a K simboli rappresenti in modo non ambiguo N elementi di **[C*]** , le sue parole devono avere una lunghezza minima. Se la parola e' lunga n, poiche' con K simboli diversi si possono realizzare K^n diverse combinazioni, per avere un codice non ambiguo deve essere rispettata la condizione $K^n > N$, cioe' n dev'essere il piu' piccolo intero che verifichi la relazione

$$n \geq \log_k N$$

Infine un codice si dice efficiente se le sue parole hanno lunghezza l = n, ridondante se l > n, ambiguo se l < n.

1.7) Codici efficienti.

Alcuni codici efficienti sono quelli usati per la rappresentazione di cifre decimali, in cui ogni cifra viene codificata indipendentemente dalle altre, mantenendosi tuttavia nell'ambito del sistema di numerazione posizionale decimale.

Poiche' per esprimere una cifra decimale sono necessari 4 bit, ma contemporaneamente con 4 bit si possono costruire 16 configurazioni diverse, 6 di esse rimangono inutilizzate e prendono il nome di configurazioni non significative.

Fra i vari codici efficienti si possono citare:

1) CODICE BCD. (BINARY CODED DECIMAL)

Le cifre 0 - 9 decimali sono codificate secondo il sistema di numerazione binario. E' questo un codice ponderato, cioe' uno di quei codici in cui ad ogni bit si puo' associare un valore, positivo o negativo, che permette di ricostruire il valore numerico rappresentato da ciascuna configurazione come somma dei pesi dei bit posti a 1.

Per tale motivo il codice BCD e' detto molto spesso anche codice 8421.

2) CODICE ECCESSO TRE

In esso la codificazione della cifra K (compresa tra 0 e 9) si fa esprimendo nel sistema di numerazione binario il numero K + 3. Non e' un codice ponderato, ma e' autocomplementante. Cio' sta ad indicare che ciascuna cifra puo' essere complementata a 9 semplicemente negando i singoli bit.

0	0 0 1 1	5	1 0 0 0
1	0 1 0 0	6	1 0 0 1
2	0 1 0 1	7	1 0 1 0
3	0 1 1 0	8	1 0 1 1
4	0 1 1 1	9	1 1 0 0

3) CODICE AIKEN O CODICE 2421

E' un codice autocomplementante e ponderato con pesi 2421.

0	0 0 0 0	5	1 0 1 1
1	0 0 0 1	6	1 1 0 0
2	0 0 1 0	7	1 1 0 1
3	0 0 1 1	8	1 1 1 0
4	0 1 0 0	9	1 1 1 1

1.8) I codici ridondanti.

Come gia' detto, i codici ridondanti usano per la codificazione degli N elementi dell'insieme **[C*]** un numero m di bit superiore a quello n strettamente necessario. La ridondanza cosi' ottenuta permette di mettere in evidenza ed eventualmente correggere gli errori di trasmissione di un messaggio. La ridondanza si ottiene aggiungendo, secondo una determinata legge, agli n bit, necessari a codificare senza ambiguita' il messaggio, k bit di controllo. Il messaggio effettivamente trasmesso contiene allora m = n + k bit. Viene chiamato ridondanza il rapporto R = m/n = = 1 + k/n.

Delle 2^m configurazioni diverse che si possono realizzare con m bit, solo 2^n sono al piu' significative, cioe' parole di codice. Le rimanenti $2^m - 2^n = 2^n .(2^k - 1)$ vengono dette configurazioni non significative.

Un errore di trasmissione viene rivelato quando trasforma una parola di codice in una configurazione non significativa; la probabilita' di rivelare un errore e' quindi tanto maggiore quanto maggiore e' la ridondanza.

E' opportuno a questo punto definire alcune quantita'.

1) Si dice **peso** il numero di bit non nulli presenti in una certa configurazione, sia essa significativa o meno. Ad esempio la configurazione 1011011 ha peso 5.

2) **Distanza** tra due configurazioni C_1 e C_2 dello stesso codice e' il numero di posizioni in cui i bit di C_1 e C_2 differiscono. Ad esempio le due configurazioni 10111 e 10100 hanno distanza 2.

3) **Molteplicita' di un errore** e' la distanza tra una configurazione C_t trasmessa e quella C_r non significativa ricevuta. Si parla quindi di errori singoli, doppi, tripli, ecc.

4) **Distanza minima di Hamming** (h) e' la minima distanza fra tutte le possibili coppie di parole di un codice. Ora gli errori sicuramente riconoscibili sono quelli che trasformano una parola P in una configurazione non significativa C_p , ma non quelli che la trasformano in una parola P'; sicuramente individuabili saranno pertanto gli errori di molteplicita' inferiore a h. I codici con h>1 sono detti rivelatori di errori. Se h e' abbastanza grande e se si suppone che C_p provenga dalla parola P che si

trova alla minor distanza, allora puo' venir effettuata la correzione dell'errore. I codici cosi' funzionanti sono detti autocorrettori.

1.8.1) Probabilita' totale di errore non rivelato.

Sia assegnata la probabilita' p, dipendente dalle caratteristiche fisiche del canale trasmissivo, che in ricezione un bit venga interpretato in modo errato. Supposto che gli errori siano tutti indipendenti tra loro, la probabilita' che r bit di una parola siano ricevuti in maniera errata mentre i rimanenti m - r siano corretti, cioe' la probabilita' che una parola P si trasformi in una configurazione C_r a distanza r da P, e':

$$P_r = p^r.(1-p)^{m-r}.\begin{pmatrix} m \\ r \end{pmatrix}$$

L'errore viene riconosciuto solo se C_r e' una configurazione non significativa; quindi se a distanza r da P si hanno N_r parole del codice, la probabilita' di avere un errore non rivelato di molteplicita' r e':

$$P_{tr} = P_{sr}.p^r.(1-p)^{m-r}.\begin{pmatrix} m \\ r \end{pmatrix}$$

avendo indicato con P_{sr} il rapporto

$$P_{sr} = \frac{N_r}{\begin{pmatrix} m \\ r \end{pmatrix}}$$

tra le parole che distano r da P e tutte le configurazioni che distano r da P.

Ne segue che la probabilita' totale di errore non rivelato e':

$$P_t = \sum_h^m N_r p^r.(1-p)^{m-r}.$$

Ora di solito p e' molto minore dell'unita' e quindi la relazione precedente puo' essere approssimata con

$$P_h = N_h.p_h$$

1.8.2) Codice a controllo di parita'.

Il codice a controllo di parita' si ottiene aggiungendo agli n bit di un codice efficiente un bit di controllo che renda pari (o dispari) il peso di ciascuna parola. Evidentemente un codice a controllo di parita' ha una distanza minima di Hamming h=2 ed e' in grado di rivelare tutti gli errori di molteplicita' dispari.

A titolo di esempio si supponga di avere a che fare con un codice di parita' da 7 bit. La ridondanza e':

$$R = 7/6 = 1,16$$

ed esistono nel codice 64 parole ed altrettante configurazioni non significative. Il numero di configurazioni che distano 2 da ciascuna parola di codice e' dato dalle combinazioni di 2 elementi su 7; tutte queste configurazioni sono evidentemente significative e pertanto parole di codice. Ne segue che $N_h = 21$.

Supponendo che il tasso di errore di bit sia p = 0,01, corrispondente ad una linea notevolmente disturbata, si ottiene:

$$P_t = N_h . p_h = 0{,}01^2 .21 = 0{,}21\ \%$$

1.8.3) Codici a peso costante.

Come dice il loro nome, sono codici a peso costante quelli in cui tutte le parole hanno lo stesso peso e per i quali quindi si ha h=2. Indicando con w il peso e con m la lunghezza della parola, il codice avra' $\binom{m}{w}$ parole, mentre le rimanenti $2^m - \binom{m}{w}$ configurazioni saranno non significative.

Il numero di parole a distanza 2 da ogni parola di codice e':

$$N_2 = w.(m - w)$$

In ogni parola infatti esistono w bit di valore 1 e (m-w) bit di valore 0. Ci sono pertanto w.(m - w) modi di scambiare un 1 con uno 0 senza alterare il peso. Si ha quindi che:

$$P_t = w.(m - w).p^2$$

Fra i codici di questo tipo particolarmente noti sono due codici decimali; il codice 2 da 5 e il biquinario. Il primo e' un codice a 5 bit con parole di peso 2

0	0 1 1 0 0	5	0 0 1 1 0
1	1 1 0 0 0	6	1 0 0 0 1
2	1 0 1 0 0	7	0 1 0 0 1
3	1 0 0 1 0	8	0 0 1 0 1
4	0 1 0 1 0	9	0 0 0 1 1

La ridondanza e':

$$R = 5/4 = 1{,}25$$

La probabilita' di errore non rivelato, sempre con p = 0,01 e':

$$P_t = 2.(5 - 2).p^2 = 0{,}06\ \%$$

Il codice biquinario e' invece un codice ad elevata ridondanza con m=7 e w=2. Sui 7 bit di ogni parola vengono effettuati due controlli per verificare che le prime due e le ultime cinque posizioni abbiano ambedue peso 1.

0	1 0 1 0 0 0 0	5	0 1 1 0 0 0 0
1	1 0 0 1 0 0 0	6	0 1 0 1 0 0 0
2	1 0 0 0 1 0 0	7	0 1 0 0 1 0 0
3	1 0 0 0 0 1 0	8	0 1 0 0 0 1 0
4	1 0 0 0 0 0 1	9	0 1 0 0 0 0 1

La ridondanza e':

$$R = 7/4 = 1,75$$

mentre la probabilita' totale di errore non rivelato e', con p = 0,01

$$P_t = 5.10^{-4} = 0,05\ \%$$

Infatti nel caso del codice biquinario $N_h = 5$ e non 10, come e' immediato verificare.

1.8.4) Codici di hamming.

I codici di Hamming sono codici con h = 3 o h = 4 usati come rivelatori d'errore o come autocorrettori. I codici con h=3 sono in grado di rivelare errori semplici e doppi, se usati come rivelatori di errore, o di correggere errori singoli, interpretando ogni configurazione non significativa come originata da quella significativa piu' vicina. In tal caso tuttavia il codice non e' in grado di rivelare gli errori doppi.

I codici con h = 4 individuano gli errori semplici, doppi e tripli quando usati come rivelatori di errore. Se usati come autocorrettori, correggono gli errori singoli e individuano quelli doppi.

In generale, se **r** e' la molteplicita' massima degli errori rilevabili e **c** quella dei correggibili, si ha:

$$r = h-1 \qquad \text{per i codici rivelatori di errori}$$

$$\begin{cases} c < \dfrac{h}{2} \\ c + r < h \end{cases} \qquad \text{per i codici autocorrerttori}$$

Il codice di Hamming con h=3 si ottiene aggiungendo agli n bit di un codice efficiente k bit di controllo ognuno dei quali sia destinato al controllo di parita' di un gruppo opportuno degli m = n + k bit risultanti.

Piu' esattamente l'i-esimo bit di controllo va situato nella posizione 2^{i-1} della parola e controlla la parita' di gruppi alternati di 2^{i-1} bit a cominciare dalla posizione 2^{i-1}.

Il primo bit di controllo va quindi nella prima posizione e controlla la parita' dei bit 1,3,5,7,9,...; il secondo bit di controllo va in posizione 2 e controlla la parita' dei bit (2,3),(6,7),(10,11),....; il terzo bit va nella quarta posizione e controlla la parita' dei bit (4,5,6,7), (12,13,14,15), e cosi' via.

In ricezione si verifica la parita' di ciascun gruppo e per ogni verifica esatta si scrive uno 0, per ogni verifica errata un 1. La stringa cosi' ottenuta, letta in senso inverso, forma il cosidetto numero di controllo N_c , che e' nullo nel caso in cui non vi siano errori. In caso

contrario ognuno dei $2^{k} - 1$ possibili valori di N_c indica in quale delle m posizioni si e' verificato un errore di molteplicita' 1.

Si supponga infatti di aver codificato con il codice di Hamming h=3 un codice efficiente da 4 bit. Si avranno pertanto 3 bit di verifica di parita', k 1 , k 2 e k 3 . Le tre verifiche di parita' eseguite in ricezione danno il seguente risultato:

k_3	Parita' dei bit	4 5 6 7	0 0 0 1 1 1 1
k_2	Parita' dei bit	2 3 6 7	0 1 1 0 0 1 1
k_1	Parita' dei bit	1 3 5 7	1 0 1 0 1 0 1
bit errato			1 2 3 4 5 6 7

Per il corretto funzionamento del codice di Hamming dev'essere quindi:

$$m \leq 2^{k} - 1$$

Si dicono ottimi quei codici per cui la relazione appena scritta e' verificata con l'uguaglianza.

Si supponga di voler codificare 16 simboli. Si ha pertanto:

$$n = 4 \qquad k \geq \log_2 (m + 1) = 3 \qquad m = 4 + 3 = 7$$

La composizione della parola, indicando con k i bit di controllo e con b i bit del codice efficiente di partenza, sara' quindi:

$$k_1 \, k_2 \, b_1 \, k_3 \, b_2 \, b_3 \, b_4$$

Il messaggio 1 0 0 1 dara' luogo alla parola di codice:

$$0\ 0\ 1\ 1\ 0\ 0\ 1$$

Si supponga che la parola ricevuta sia invece:

$$0\ 0\ 0\ 1\ 0\ 0\ 1$$

I primi due controlli di parita' (sui bit 1,3,5,7 e (2,3) e (6,7) rispettivamente) danno risultato errato, mentre il terzo controllo (sui bit 4,5,6,7) e' esatto. Il numero di controllo e' quindi:

$$N_c = 0\ 1\ 1$$

corrispondente alla posizione del bit errato. La ridondanza del codice e' R = 7/4 = 1,75

Per quanto riguarda la probabilita' di errore non rivelato, anche ammettendo che tutte le configurazioni a distanza 3 da una parola del codice siano esse stesse parole del codice, si ha con p = 0,01:

$$P_t \leq 15.0{,}01^3 = 0{,}0015\ \%$$

I codici di Hamming con h=4 si ottengono da quelli con h=3 aggiungendo in ultima posizione un ulteriore bit di controllo che verifichi la parita' di tutti i bit che lo precedono.

La verifica di un messaggio si fa controllando tutti i bit di parita'. Se i primi (k - 1) controlli non sono tutti esatti mentre l'ultimo lo e' allora si e' in presenza di un errore doppio non correggibile.

Supponendo di aver ottenuto dal codice h=3 appena illustrato quello h=4, si ha:

$$R = 8/4 = 2 \qquad\qquad P^t \leq 15.p^4 = 1{,}5.10^{-7}$$

1.9) Codici riflessi.

La caratteristica principale dei codici riflessi, detti anche codici ciclici, e' che ogni configurazione significativa differisce dalla precedente e dalla seguente per un solo bit.

Sono largamente usati nei convertitori analogico-digitali, mentre non vengono usati in trasmissione, essendo h=1, ne' nei calcolatori non essendo ponderati. Un codice ciclico si dice completo in n variabili se contiene in sequenza ciclica tutte le 2^n combinazioni delle variabili, incompleto in caso contrario. Uno dei piu' diffusi codici riflessi e' quello di Gray. Dal codice G_n di Gray a n bit si puo' ricavare quello G_{n+1} a n+1 bit con il seguente procedimento:

1) Al numero decimale $2^n + k$ si assegna la stessa configurazione di bit del numero $2^n - (k + 1)$ per $k = 0,1,2,.....,2^{n-1}$.

n = 1	**n = 2**	**n = 3**	**n = 4**
0	0 0	0 0 0	0 0 0 0
1	0 1	0 0 1	0 0 0 1
	1 1	0 1 1	0 0 1 1
	1 0	0 1 0	0 0 1 0
		1 1 0	0 1 1 0
		1 1 1	0 1 1 1
		1 0 1	0 1 0 1
		1 0 0	0 1 0 0
			1 1 0 0
			1 1 0 1
			1 1 1 1
			1 1 1 0
			1 0 1 1
			1 0 1 0
			1 0 0 1
			1 0 0 0

2) Si premette a tutti i numeri $< 2^n$ uno 0 e a tutti quelli $\geq 2^n$ un 1. In pratica basta eseguire un specularita' del codice a n bit, come illustrato nello schema soprastante, e premettere le cifre 0 e 1 alle due immagini cosi' ottenute.

Esistono delle semplici regole di conversione di un parola P_G del codice Gray alla parola P_B corrispondente in numerazione posizionale binaria e viceversa.

Nel primo caso:

1) Si procede da sinistra verso destra: fino al primo bit 1 di P_G, $P_B = P_G$.

Successivamente:

2) Quando l'i-esimo bit di $P_G = 0$, l'i-esimo bit di P_B e' uguale a quello (i-1)-esimo di P_B .

3) Quando l'i-esimo bit di $P_G = 1$, l'i-esimo bit di P_B e' la negazione di quello (i-1)-esimo di P_B .

Le regole per la conversione inversa da binario a codice Gray sono altrettanto semplici:

1) Si procede da sinistra verso destra confrontando l'i-esimo bit di P_B con l'(i-1)- esimo. Se i due bit sono uguali, l'i-esimo bit di P_G e' 0; se sono diversi e' 1.

2) Il primo bit si confronta con 0.

In fig. 1.9.1 sono riportati gli esempi sia della conversione da Gray a binario che da binario a Gray.

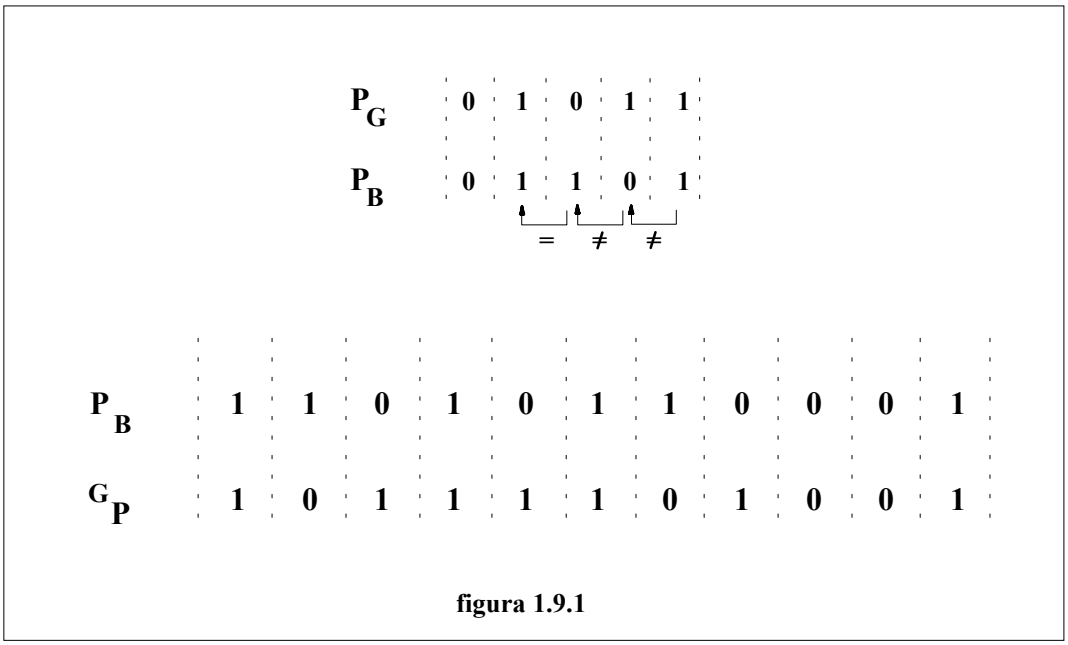

figura 1.9.1