Il coefficiente di Ridondanza è R = Numero di combinazioni possibili / Numero di combinazioni usate. Esso è sempre maggiore o uguale a 1 e misura l’efficienza della nostra codifica. Se R è minore di 2 il codice è detto **Minimale.** Se il codice non è minimale diventa possibile l’utilizzo di una rappresentazione diversa è più efficiente, ma ciò non sempre è necessario/comodo.

Esempio: se si vuole rappresentare un alfabeto di 65 numeri, si devono utilizzare almeno 7 bit. In quel caso il coefficiente di Rid = 128 / 65 (che è molto vicino a 2).

Se il codice è a lunghezza variabile (come la scrittura di numeri su un foglio), però, la situazione cambia. Introduciamo per esempio la numerazione romana; essa è composta da 7 simboli di base (I V X L C D M = 1 5 10 50 100 500 1000). I numeri diversi da quelli di base vengono costruiti tramite somme e sottrazioni tra i simboli: Per esempio 3 viene scritto III, 6 diventa VI. Questo fa sì che il numero di cifre necessarie non cresca linearmente con l’aumentare del valore (3 è più piccolo di 6 ma richiede più cifre). Un altro esempio è il 4 che può essere rappresentato sia come IIII che come IV: in questo caso esistono due modi di scrivere uno stesso numero che occupano numeri di cifre parecchio differenti (2 e 4). Si supporrebbe che si utilizzi sempre la rappresentazione che occupa meno cifre, ma a causa delle limitazioni sulla scrittura dei numeri romani 8 viene scritto VIII anziché IIX. Il numero 149 poi, anziché essere rappresentato CIL, è CXLIX (si separano centinaia C, decine XL e unità IX).

Se si passa a un supporto informatico per poter rappresentare un codice a lunghezza variabile è necessario specificare il numero di cifre che vengono utilizzate: quindi è necessario almeno un carattere --di terminazione-- in più che indichi la fine del numero -> ad esempio si potrebbe utilizzare il punto. Aggiungendo il simbolo di stop i simboli utilizzabili diventerebbero 8 (quindi sarebbe possibile rappresentarli con 3 bit). Si possono infatti mappare così:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | . | stop |
| 0 | 0 | 1 | I | 1 |
| 0 | 1 | 0 | V | 5 |
| 0 | 1 | 1 | X | 10 |
| 1 | 0 | 0 | L | 50 |
| 1 | 0 | 1 | C | 100 |
| 1 | 1 | 0 | D | 500 |
| 1 | 1 | 1 | M | 1000 |

E si potrebbero cambiare le regole per inventare una codifica “genovese” che usi al meglio il numero di bit.

8 = IIX. -> 001001011000

0 = . -> 000

Confrontiamo adesso questo tipo di rappresentazione con la rappresentazione binaria a lunghezza fissa. Nella rappresentazione binaria prendiamo per esempio 10 bit (che permette di arrivare fino a 1023).

Per scrivere i valori 0, 1, 2 nella rappresentazione “genovese” si occupano 3,6,9 bit (meno di 10). Così accade anche coi valori 4, 5, 6 (9, 6, 9 bit, si risparmia almeno 1 bit). Tuttavia già il valore 3, che richiede tre cifre più terminatore occupa 12 bit nella rappresentazione “genovese”. Il valore 2000 (che è maggiore di 1024) è possibile scriverlo come MM. (occupando 9 bit). Tuttavia, specialmente coi valori a più cifre non multipli di 5, il numero di bit continua a crescere. Perciò, sebbene un tipo di rappresentazione permetta di risparmiare qualche bit in determinati casi, essi sono puramente fortuiti. Le applicazioni di compressione dei file (Zip / Rar) vanno a guardare il numero di occorrenze dei vari casi nei file che analizzano e li convertono in un tipo di rappresentazione a lunghezza variabile che varia in funzione della frequenza di occorrenze dei vari “numeri” (se ci sono molte occorrenze del valore 5 per esempio, utilizzeranno una rappresentazione a lunghezza variabile simile a quella “genovese” che premi in risparmio di bit l’occorrenza di questi valori).

Stringa di caratteri (appartenenti alla tabella ASCII a 7 bit). Il terminatore di stringa è lo 0 (0000000) che non è stampabile. Se io scrivo una frase del tipo “La mia cara mamma cucina molto bene” (la cui codifica terminerà con 0000000) la sua lunghezza è 7 bit \* il numero di caratteri (spazi compresi + il bit di terminazione), cioè 7 \* 36 bit = 252 bit.

Se si volesse comprimere questa stringa si potrebbe costruire una tabella ridotta che non contenga i caratteri mancanti. Infatti, nella stringa sono presenti solo: a, b, c, e, i, l, m, n, o, r, t, u, spazio, terminatore (ossia 14 caratteri). Si potrebbe quindi creare una tabella di soli 14 caratteri e codificarli con una rappresentazione a lunghezza fissa a 4 bit. Un’altra opzione sarebbe contare il numero di volte che si presentano i vari caratteri nella stringa (a:7, b:1, c:3, e:2, i:2, l:2, m:5, n:2, o:2, r:1, t:1, u:1, spazio:6, terminatore:1). Si potrebbe quindi creare un codice a lunghezza variabile che sfrutti il fatto che alcuni caratteri si presentino molto più frequentemente degli altri. In particolare, si può utilizzare un codice che sia una via di mezzo tra lunghezza fissa e variabile, ossia un codice a espansione. Lo si fa partire da 2 bit (00, 01, 10, 11) di cui 3 combinazioni codifichino i caratteri a, spazio ed m (00:a, 01:spazio, 10:m 11:espansione). 11 viene utilizzato per espandere il codice a, per esempio, 4 bit: ciò permette di aumentare il numero di valori codificabili. Aumentato a 4 bit si possono prendere quindi anche c, e, i, (1100:c, 1101:e, 1110:c, 1111:espansione). In questo caso si porta il codice a 7 bit (3 bit di espansione anziché 2) e ciò permette di codificare anche i caratteri mancanti. Ci sono quindi 3 livelli di espansione: 2, 4 e 7 bit.

La codifica di partenza prevedeva l’utilizzo di 7 bit in totale per ciascun carattere, arrivando a un totale di 252 bit.

Con l’ipotizzata tabella a 4 bit per carattere, il codice avrebbe occupato 144 bit.

Utilizzando la codifica a lunghezza espandibile (codice a espansione), il numero di bit occupati sarebbero 7\*2 + 1\*7 + 3\*4 + 2\*4 + 2\*4 + 2\*7 + 5\*2 + 2\*7 + 2\*7 + 1\*7 + 1\*7 + 1\*7 + 6\*2 + 1\*7 = 141 bit.

Per poter effettuare una compressione (che sia utile) è necessaria quindi una distribuzione Poco uniforme dei valori; per esempio, una stringa generata da un generatore di lettere casuali avrà una quantità quasi omogenea di ciascuna lettera e perciò non si appresterà bene alla compressione.

Riassumendo, abbiamo visto i codici a lunghezza fissa (in cui i dati occupano sempre lo stesso numero di bit, sono efficienti ma hanno limitazioni), i codici a lunghezza variabili (che occupano numero di bit variabile e non avendo limitazioni possono Potenzialmente rappresentare qualsiasi cosa, ma possono rivelarsi inefficienti in molti casi non particolari) e i codici a espansione (che sono una via di mezzo, e tornano utili per le operazioni di compressione di dati specifici).

Quasi sempre le istruzioni in linguaggio macchina nei sistemi di calcolo sono rappresentate utilizzando un codice ad espansione.