Corso di Calcolatori Elettronici I

Informazione e sua rappresentazione: codifica

ing. Alessandro Cilardo

Corso di Laurea in Ingegneria Biomedica

Il concetto di informazione

- Qualunque informazione è definita tramite tre caratteristiche fondamentali:
  1. **Valore**
     - indica il particolare elemento assunto dall'informazione
  2. **Tipo**
     - indica l'insieme degli elementi entro cui è stato scelto il valore attribuito all'informazione
  3. **Attributo**
     - indica il significato associato all'informazione nel contesto in cui questa viene utilizzata
- Si ottiene un'informazione completa quando un attributo assume un valore di un determinato tipo
- Fornire un'informazione significa effettuare la scelta di un elemento in un insieme definito di oggetti

Informazione: esempio

- **Attributo**:
  Soluzione dell'equazione di primo grado
- **Valore:**
  3,45
- **Tipo**:
  Numeri reali

Differenza tra valore e rappresentazione del valore

- Non confondere il valore dell'informazione (*elemento di un insieme*) con la sua rappresentazione
  - 3 III tre
  - padre pere father
  - 0.1 1/10 1*10E(-1)

Tipi

- Semplici
  - I valori sono entità atomiche
    - Es: tipo reale `float f;`
- Strutturati
  - I valori sono aggregati di informazioni più semplici
    - Es: Dati Anagrafici

```
struct persona {
        char    name[SIZENAME];
        char    tfnumb[SIZETELE];
        char    addr[SIZEADDR];
        struct persona *ptrnext;
};
```

Cardinalità di un tipo

- Ogni tipo ha una propria cardinalità *N* che è pari al numero di elementi che compongono il tipo
- La ***cardinalità*** esprime il numero di elementi tra cui scegliere; essa può essere usata per misurare la quantità di informazione
- Una scelta fra valori di un tipo di cardinalità *N* é più complessa di una scelta fra valori di un tipo di cardinalità *M*, se *N*>*M*
  - il tipo di cardinalità *N* ha una quantità di informazione maggiore

Il bit

- L'informazione più elementare è legata alla scelta tra due oggetti
- Un'informazione il cui tipo ha cardinalità 2 è detta un ***bit (binary digit)***
- Il valore di un tipo di cardinalità N può essere rappresentato come un insieme ordinato di k bit

$$2^k \geq N \qquad\qquad k = \lceil \log_2 N \rceil$$

  - il simbolo $\lceil\ \rceil$ indica la funzione *ceiling* che restituisce il primo intero maggiore o uguale all'argomento
- Nei calcolatori elettronici le informazioni sono rappresentate come stringhe di bit

Il bit come unità di informazione

- E' possibile stabilire per una informazione il cui tipo sia a cardinalità N a quanti bit equivale la quantità di informazione ad essa associata
- Il problema può essere posto in questi termini:

  a quante scelte fra 2 equivale una scelta fra N?

Esempi

- Tipo: **ColoreSem{verde,rosso,giallo}** cardinalità=**3**
- A quanti bit equivale la quantità di informazione ad esso associata? **B=$\lceil \log_2 3 \rceil$=2** ($3<2^2$)

- Tipo: **Cifre{0,1,2,3,4,5,6,7,8,9}** cardinalità=**10**
- A quanti bit equivale la quantità di informazione ad esso associata? **B=$\lceil \log_2 10 \rceil$=4** ($10<2^4$)

- Tipo: **Mesi{gen,feb,mar,apr,mag,giu,lug,ago,set,ott,nov,dic}** cardinalità=**12**
- A quanti bit equivale la quantità di informazione ad esso associata? **B=$\lceil \log_2 12 \rceil$=4** ($12<2^4$)

Memorizzazione dell'informazione

- Un calcolatore opera su DATI, cioè sulla rappresentazione mediante codifica del valore dell'informazione
- Tale rappresentazione è memorizzata in organi di memoria, i REGISTRI, che possono assumere un numero **finito** di configurazioni distinte
- La cella elementare di memoria è un elemento fisico bi-stabile detto flip-flop, atto a memorizzare il valore di un bit di informazione

Codifica delle informazioni

- *Rappresentazione* del valore di una informazione di tipo $D = (x_1,\ldots,x_N)$ (*alfabeto origine*) mediante stringhe di simboli di un tipo $R = (a_1,\ldots,a_k)$ (*alfabeto codice*)
- Funzione *iniettiva* dall'insieme *D* all'insieme $C = R^l$

$$c : D \rightarrow C = R^l = R \times R \times \ldots \times R$$

$$|R^l| \geq |D|$$

- Tabella Codice: tabella che associa a ciascun elemento $x_i \in D$ una stringa di lunghezza l_i di elementi di R, detta *parola codice*

Codifica a lunghezza fissa

- $l_i = l$ = costante per tutti gli elementi di D
- Per codificare un tipo di cardinalità N mediante un alfabeto di k simboli è necessaria una stringa di lunghezza l tale che sia possibile far corrispondere a ciascun elemento $x_i \in D$ una distinta tra le k^l disposizioni con ripetizione dei k simboli di R sugli l posti della stringa

$$\mathbf{k^l \geq N} \qquad \text{da cui} \qquad \mathbf{l \geq \lceil \log_k N \rceil}$$

- Se $l = m = \lceil \log_k N \rceil$ il codice si dice *a lunghezza minima*

Codici completi ed incompleti

- se **l = m = $\log_k$ N, con N potenza di k**,
  il codice viene detto ***completo***
- se **l = m** (codice a lunghezza minima)
  ma N non è potenza di k,
  il codice viene detto ***incompleto***
- la differenza **k^m - N** fornisce il numero di parole codice non assegnate, cioè non associate ad alcun elemento dell'alfabeto origine
- Se k = 2 ed N = 65, allora m = 7
  di tutte le potenziali 2^7 = 128 stringhe di 7 bit, solo 65 sono parole codice e le rimanenti 128-65 = 63 sono non assegnate

Codici ridondanti

- Se **$l > m = \lceil \log_k N \rceil$**,
  il codice viene detto ***ridondante***
- I codici ridondanti vengono utilizzati per rilevare ed eventualmente correggere errori dovuti ad alterazioni del dato
- Una misura della ridondanza è:

$$r = 1 - [(\log_k N)/l] \in [0,1]$$

- r è uguale a 0 per $l = \log_k N$ (codice completo)
  r tende ad 1 per l che tende ad infinito

Codici ridondanti

$k^m \geq N$

N = 8, K = 2	→	m = 3, l = 3, (l = m)	→ Codice Completo & Non Ridondante
N = 7, K = 2	→	m = 3, l = 3, (l = m)	→ Codice InCompleto & Non Ridondante
N = 8, K = 2	→	m = 3, l = 4, (l > m)	→ Codice InCompleto & Ridondante

Codifica a lunghezza variabile

- La lunghezza l_i della parola codice dipende da x_i :

$$l_i = f(x_i)$$

- Proprietà fondamentale è quella che ogni parola codice non si ritrova come sequenza iniziale (prefisso) di altre parole codice più lunghe
- L'uso di questo tipo di codifica è giustificato quando gli elementi del tipo D (alfabeto origine) non hanno tutti la stessa probabilità di occorrenza
  - Parole codice più corte associate a elementi dell'alfabeto origine con maggiore probabilità di occorrenza

Codifica a lunghezza variabile

- Dato l'alfabeto sorgente D = (x_1,…..,x_n), dette p_1,…..,p_n le probabilità di occorrenza (frequenza) dei rispettivi elementi di D, la lunghezza l_i viene scelta in modo da minimizzare la *lunghezza media* L_m del codice:

$$L_m = \Sigma_{i=1,n} (p_i * l_i)$$

- Effettuata la ricerca della n-pla di valori l_i che rende minima L_m si ottiene un codice a lunghezza variabile a minima ridondanza
- La medesima informazione codificata con un codice a lunghezza fissa richiede un codice di lunghezza maggiore di L_m

Esempi di codici

- Codici a lunghezza fissa
  - Codice Fiscale
  - Codice di Avviamento Postale
- Codici a lunghezza variabile
  - Alfabeto Morse
  - Numeri Telefonici

Codifica a lunghezza variabile

- Data la rappresentazione $c : D \rightarrow C$, l'insieme C può essere costituito da stringhe di lunghezza differente
- Esempio (stringhe di cifre da 0 a 3):

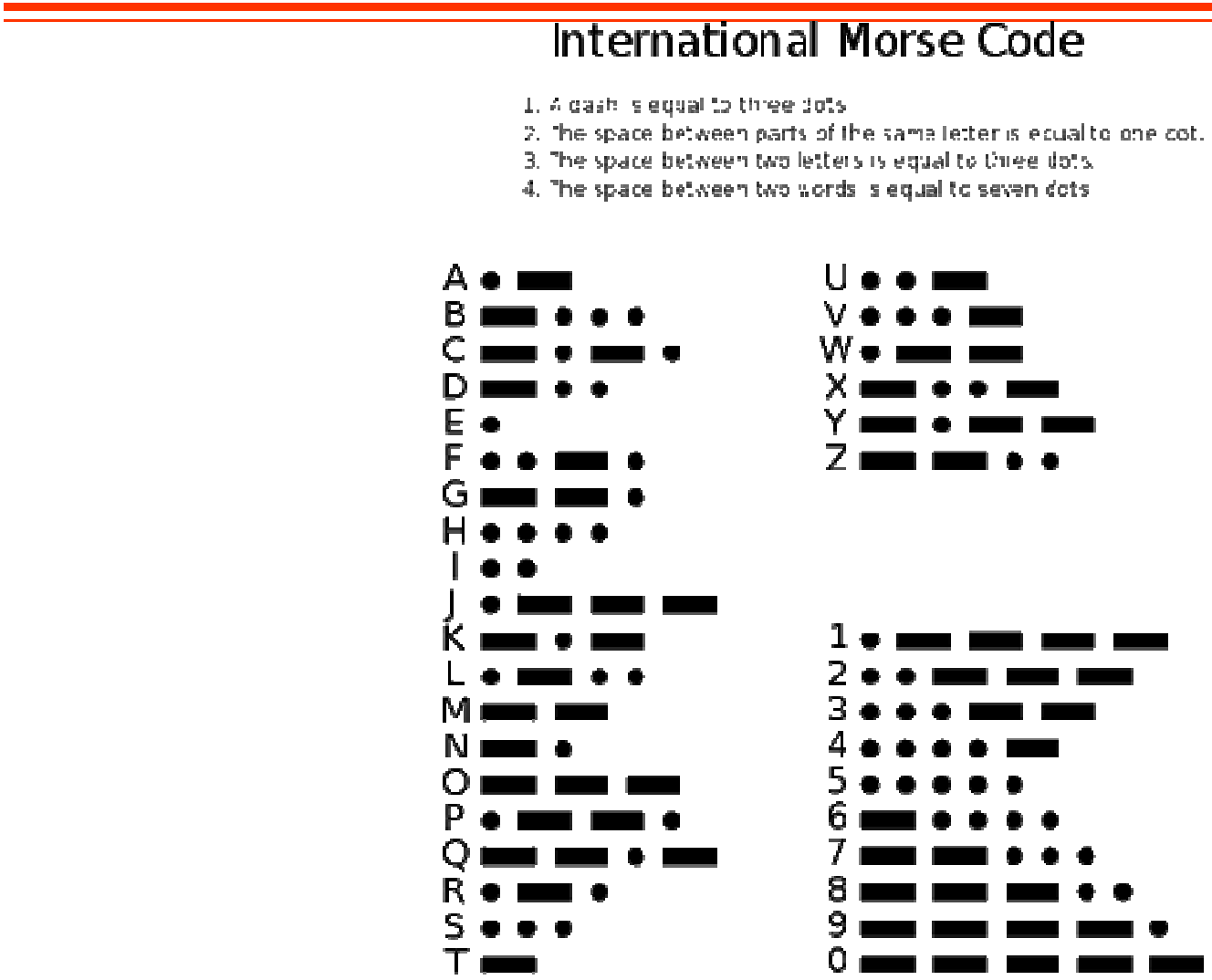
Informazione	*codice*	*Informazione*	*codice*	*Informazione*	*codice*
Casa	0	Banca	32	Andrea (C)	3310
Genitori	1	Paolo	3300	Andreani	3311
Segretaria	2	Anna	3301	Marocco	3312
Direttore	30	Mario	3302	Daita	3313
Taxi	31	Mario (C)	3303	Luchini	3320

Codifica a lunghezza variabile

- La corrispondenza viene decisa tenendo conto della frequenza con cui vengono usati i valori in *D*

- Vantaggi:
  - Risparmio di spazio nella memorizzazione
  - Risparmio di tempo nella trasmissione

Esempio: codice Morse



Esempio di rappresentazione di codici mediante tabella: BCD

Codice BCD (Binary Coded Decimal)

$D = (x_1, x_2, \ldots, x_N) = (0,1,2,\ldots,9)$ alfabeto origine

$R = (a_1, a_2, \ldots, a_k) = (0,1)$ alfabeto codice

$l = m = 4$

	0	1	2	3
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

Parola Codice

Codifica in Binario

- Dato un insieme di N elementi, il numero minimo l di bit necessario alla codifica è:

$$l = \lceil \log_2 N \rceil$$

Rappresentazione Decodificata

- Codice ridondante in cui le parole codice hanno lunghezza l = N
- Ogni parola codice contiene un solo bit di valore 1 ed N-1 bit di valore 0
- *Es: N=4*
  - *Cuori* $\rightarrow$ *1000*
  - *Quadri* $\rightarrow$ *0100*
  - *Fiori* $\rightarrow$ *0010*
  - *Picche* $\rightarrow$ *0001*

Esempio di rappresentazione decodificata

Dato	Codice BCD	Codice Decodificato
0	0000	1000000000
1	0001	0100000000
2	0010	0010000000
3	0011	0001000000
…	…	….
…		
9	1001	0000000001

Codifica indiretta

Codifica diretta

$D=(x_1,x_2,\ldots,x_{20})$

$R=(0,1)$

$$m = \lceil \log_2 N \rceil$$

Codifica indiretta

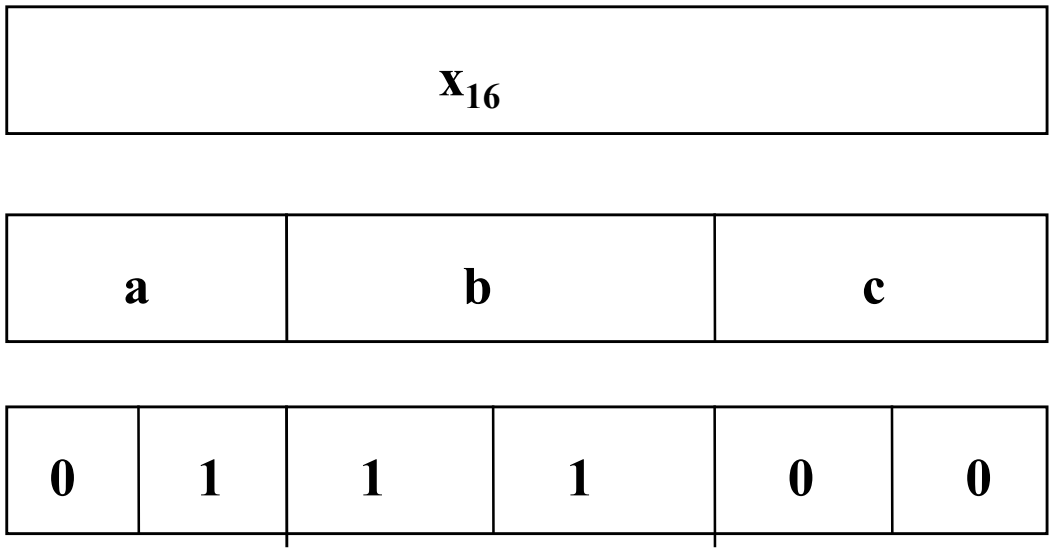
$D=(x_1,x_2,\ldots,x_{20})$ alfabeto origine

$J=(a,b,c)$ alfabeto intermedio,
di cardinalità k intermedia $2<k<20$

$R=(0,1)$ alfabeto codice

$$X_{16} \rightarrow abc \rightarrow 011100$$

Codifica Indiretta



$D=(x_1,x_2,\ldots,x_{20})$ alfabeto origine

$J=(a,b,c)$ alfabeto intermedio

$R=(0,1)$ alfabeto codice

La lunghezza della parola codice è di 6 bit, mentre con una codifica diretta la lunghezza minima è di 5 bit

Rappresentazione dei caratteri

- E’ necessario poter rappresentare dati di tipo carattere
- Il tipo carattere contiene non solo le lettere dell’alfabeto (quale?!?) ma anche altri caratteri: cifre decimali, segni di interpunzione, caratteri speciali, di controllo etc.
- Qual è la cardinalità del tipo carattere?

Rappresentazione dei caratteri

- D=caratteri, R=cifre binarie
- Se m=7, allora $|D|=2^7=128$
- Se m=8, allora $|D|=2^8=256$

- Codici standard sono stati proposti allo scopo di facilitare lo scambio di testi codificati tra diversi sistemi di calcolo, ad esempio: ASCII, UNICODE, etc…

Codice ASCII

- ASCII=*American Standard Code for Information Interchange*
- Nella sua versione originale prevede di rappresentare 128 caratteri con stringhe di 7 bit
- I caratteri sono elencati seguendo un particolare ordine e associati a stringhe di bit corrispondenti a numeri naturali crescenti da 0 a 127 (se interpretate come numeri binari)
- Poiché i registri di un calcolatore moderno contengono un numero di bit che è un multiplo di 8, per rispettare il codice ASCII si pone l'ottavo bit (quello più a sinistra) pari a 0

Codice ASCII

- Schema (notare che in questo modo si introduce un ordinamento relativo):
  - da 0 a 31: “caratteri” di controllo
  - da 32 a 47: interpunzione e caratteri speciali
  - da 48 a 57: cifre decimali
  - da 58 a 64: interpunzione e caratteri speciali
  - da 65 a 90: lettere maiuscole dell’alfabeto inglese
  - da 91 a 96: interpunzione e caratteri speciali
  - da 97 a 122: lettere minuscole alfabeto inglese
  - da 123 a 127: caratteri speciali
- ASCII(‘a’)-ASCII(‘A’) = … = ASCII(‘z’)-ASCII(‘Z’) = 32

Codice ASCII: tabella

Byte	Cod.	Char
00000000	0	Null
00000001	1	Start of heading
00000010	2	Start of text
00000011	3	End of text
00000100	4	End of transmit
00000101	5	Enquiry
00000110	6	Acknowledge
00000111	7	Audible bell
00001000	8	Backspace
00001001	9	Horizontal tab
00001010	10	Line feed
00001011	11	Vertical tab
00001100	12	Form Feed
00001101	13	Carriage return
00001110	14	Shift out
00001111	15	Shift in
00010000	16	Data link escape
00010001	17	Device control 1
00010010	18	Device control 2
00010011	19	Device control 3
00010100	20	Device control 4
00010101	21	Neg. acknowledge
00010110	22	Synchronous idle
00010111	23	End trans. block
00011000	24	Cancel
00011001	25	End of medium
00011010	26	Substitution
00011011	27	Escape
00011100	28	File separator
00011101	29	Group separator
00011110	30	Record Separator
00011111	31	Unit separator

Byte	Cod.	Char
00100000	32	Spc
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?

Byte	Cod.	Char
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_

Byte	Cod.	Char
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	Del

Estensioni del codice ASCII

- Utilizzando l'ottavo bit
  - i caratteri da 128 a 255 rappresentano vari caratteri speciali, simboli matematici e lettere non appartenenti all'alfabeto inglese
  - Questi codici non sono altrettanto universali
  - Non sempre sono correttamente interpretati

Standard UNICODE

- E’ uno standard recente
- $|D|=2^{16}$ (ma anche più di 16 bit)
- Risolve in maniera completa il problema della specificità delle lingue e dei simboli matematici
- E’ compatibile con ASCII (lo contiene)