Rappresentazione dei numeri naturali

Prof. Daniele Gorla

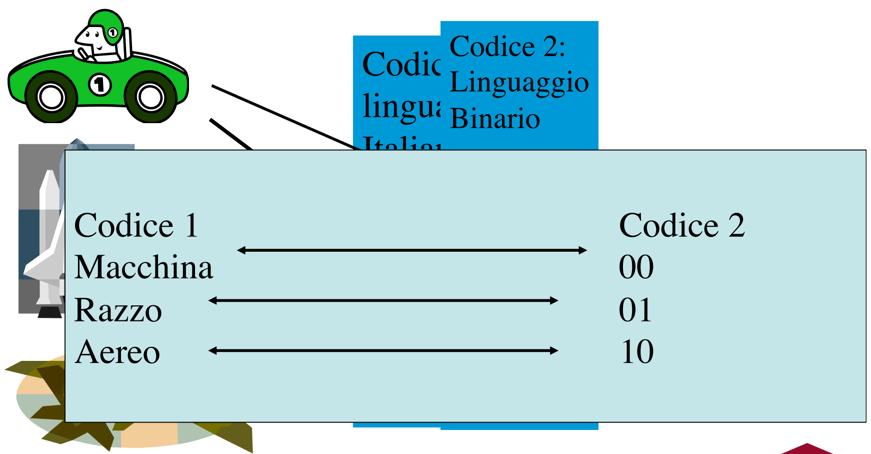
Rappresentazione dell'Informazione

I calcolatori elettronici sono macchine in grado di elaborare informazioni trasformandole in altre informazioni.

Nel mondo dell'informatica, intendiamo in modo più restrittivo per informazione *tutto ciò che può essere rappresentato tramite opportune sequenze di simboli in un alfabeto prefissato*.

Un *codice* C è un insieme di parole composte da simboli di un alfabeto Σ (detto *alfabeto di supporto* di C).

Esempi di codici



Codifica e decodifica

CODIFICA

La *codifica* di un insieme di informazioni I in un dato codice C è una funzione

$$f : \mathrm{I} \rightarrow \mathrm{C}$$

Esempio:

macchina → *00* *razzo* → *01* *aereo* → *10*

dove: I è un sottoinsieme di parole della lingua italiana

C è un sottoinsieme delle parole composte da 0 e 1

DECODIFICA

La *decodifica* di una informazione codificata in precedenza è una corrispondenza

$$g : \mathrm{C} \rightarrow \mathrm{I}$$

(tipicamente, è la funzione inversa di f)

Esempio:

00 → *macchina* *01* → *razzo* *10* → *aereo*

Criteri di valutazione di una codifica

Economicità: sono considerate migliori rispetto a questa caratteristica le codifiche che utilizzano pochi simboli.

Semplicità di codifica e decodifica: è auspicabile poter trasformare un linguaggio da un codice all'altro in modo efficiente

Semplicità di elaborazione: sono preferibili le codifiche che consentono di eseguire le operazioni definite sui dati in modo agevole (ad esempio, sostituendo ai simboli arabi i simboli dei numeri romani, "saltano" il meccanismo del riporto e della posizionalità).

Sistemi posizionali

Un sistema numerico *posizionale* in base *b*, ovvero basato su un **alfabeto** Σ di *b* simboli distinti, consente di esprimere un qualsiasi numero naturale *N* di *m* cifre, mediante la formula

$$N = \sum_{i=0}^{m-1} c_i b^i, \quad c_i \in \Sigma$$

Ad esempio, nel sistema decimale (b=10, Σ=0,1,..9), la sequenza $N_{10} = 254$ esprime il numero

$$254_{10} = 200 + 50 + 4$$
$$= 2 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0$$

Similmente, in base 7 (b=7, Σ=0,1,..6), la stessa sequenza di cifre $N_7 = 254$ esprime il numero

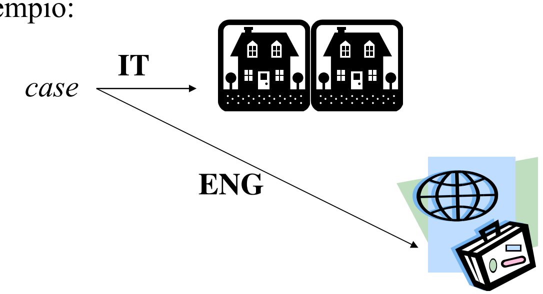
$$254_7 = 2 \cdot 7^2 + 5 \cdot 7^1 + 4 \cdot 7^0 \; (= 137_{10})$$

Quindi ad esempio 10_{10} e 10_2 hanno significato **diverso** anche se le sequenze di simboli sono le stesse

(1 seguito da 0)!!

Bisogna **interpretare** una stringa di simboli, una volta che ci venga detto il codice utilizzato per generarla (*decodifica*)

Esempio:



	Base 2	Base 3	Base 4	Base 5	...	Base 8	...	Base 10	...	Base 16
	0000	000	00	00		00		00		0
	0001	001	01	01		01		01		1
	0010	002	02	02		02		02		2
	0011	010	03	03		03		03		3
	0100	011	10	04		04		04		4
	0101	012	11	10		05		05		5
	0110	020	12	11		06		06		6
N_b	0111	021	13	12		07		07		7
	1000	022	20	13		10		08		8
	1001	100	21	14		11		09		9
	1010	101	22	20		12		10		A
	1011	102	23	21		13		11		B
	1100	110	30	22		14		12		C
	1101	111	31	23		15		13		D
	1110	112	32	24		16		14		E
	1111	120	33	30		17		15		F

Codice binario

SAPIENZA Università di Roma Dipartimento di Informatica

Codice binario: un codice costituito dai soli simboli 0 ed 1.

Quindi, b=2 e $\Sigma = \{0,1\}$

I simboli 0 ed 1 prendono il nome di *bit*, una contrazione per *binary digit*.

Perchè il codice binario?

- *George Boole* dimostrò come la logica possa essere ridotta ad un sistema algebrico molto semplice che utilizza solo un codice binario.
- Il codice binario fu trovato particolarmente utile nella *teoria della commutazione* (Claude Shannon) per descrivere il comportamento dei circuiti digitali (1=acceso, 0=spento).

Ogni tipo di informazione può essere codificata con un codice binario.

- Cominciamo con i numeri: i codici saranno **diversi** a seconda che si vogliano rappresentare numeri naturali, interi, razionali, …
- Si possono poi rappresentare parole (sequenze di caratteri dell'alfabeto – codici ASCII e UNICODE), immagini, suoni, …

Rappresentazione binaria di numeri naturali

SAPIENZA Università di Roma Dipartimento di Informatica

Un **numero naturale** N in base 2 di n bit può essere rappresentato mediante la formula:

$$\boxed{N_2 = \sum_{i=0}^{n-1} c_i 2^i,\ c_i \in \{0,1\}}$$

Il bit c_0 viene detto LSB (*less signifying bit*) mentre c_{n-1} viene detto MSB (*most signifying bit*).

Esempio: $111001_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

Cambiamento di base per i numeri naturali

SAPIENZA Università di Roma Dipartimento di Informatica

Problema: convertire un numero N espresso in base a (N_a) in un numero N' espresso in base b (N'_b)

Conversione di base: metodo polinomiale

SAPIENZA Università di Roma Dipartimento di Informatica

Si usa l'espressione

$$N_a = \sum_{i=0}^{n-1} c_i a^i,\ c_i \in \{0,\dots,a-1\}$$

Si esprime il numero N_a come un polinomio, **usando i numeri dell'alfabeto *b*** nel polinomio

Si valuta il polinomio **usando l'aritmetica in base *b***

Esempio (da base 2 a base 10):

$$111001_2 = (1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0)_{10}$$
$$= (32+16+8+0+0+1)_{10} = 57_{10}$$

Metodo polinomiale

Difficile da usare se la base di arrivo (b) non è 10 (bisogna lavorare nell'aritmetica modulo b!!)

Esempio (da base 7 a base 3):
$3602_7 = (10 \cdot 21^3 + 20 \cdot 21^2 + 0 \cdot 21^1 + 2 \cdot 21^0)_3$

SERVE UN ALTRO METODO!!!

Teorema della divisione euclidea

Per ogni $D,d \in \mathbf{N}$, esiste un'unica coppia $q,r \in \mathbf{N}$ t.c.

$$\underbrace{D}_{\text{dividendo}} = \underbrace{q}_{\text{quoziente}} \cdot \underbrace{d}_{\text{divisore}} + \underbrace{r}_{\text{resto}}, \quad \text{con } 0 \leq r < d$$

Notazionalmente, $q = D \text{ div } d$
$r = D \text{ mod } d$

Per esempio, 7 div 3 = 2
7 mod 3 = 1

$$N = c_0 + c_1 \cdot b^1 + c_2 \cdot b^2 + c_3 \cdot b^3 + \ldots + c_{n-2} \cdot b^{n-2} + c_{n-1} \cdot b^{n-1}$$
$$= \underbrace{c_0}_{\text{resto}} + \underbrace{b}_{\text{divisore}} \cdot \underbrace{(c_1 + c_2 \cdot b^1 + c_3 \cdot b^2 + \ldots + c_{n-2} \cdot b^{n-3} + c_{n-1} \cdot b^{n-2})}_{\text{quoziente}}$$

dividendo

da cui

$N \text{ mod } b = c_0$

$N \text{ div } b = c_1 + c_2 \cdot b^1 + c_3 \cdot b^2 + \ldots + c_{n-2} \cdot b^{n-3} + c_{n-1} \cdot b^{n-2} = N^{(1)}$

Iterando

$N^{(1)} \text{ mod } b = c_1$

$N^{(1)} \text{ div } b = c_2 + c_3 \cdot b^1 + \ldots + c_{n-2} \cdot b^{n-4} + c_{n-1} \cdot b^{n-3} = N^{(2)}$

$N^{(2)} \text{ mod } b = c_2$

$N^{(2)} \text{ div } b = c_3 + \ldots + c_{n-2} \cdot b^{n-5} + c_{n-1} \cdot b^{n-4} = N^{(3)}$

…

Finché arrivo ad $N^{(n-1)} = c_{n-1}$ e allora avrò

$N^{(n-1)} \text{ mod } b = c_{n-1}$

$N^{(n-1)} \text{ div } b = 0$

Conversione di base: metodo delle divisioni iterate

Convertire N_a in base b

1. Dividi ripetutamente N_a per b_a
   (N.B.: ***b* va espresso in base *a* e la divisione va fatta in base *a***)
2. I resti delle divisioni convertiti in base b danno le cifre, dalla meno significativa alla più significativa, di N_a espresso in base b.

Esempio: 657_{10} in base 4

657 : 4 = 164 resto 1
164 : 4 = 41 resto 0
41 : 4 = 10 resto 1
10 : 4 = 2 resto 2
2 : 4 = 0 resto 2

Quindi, $657_{10} = 22101_4$

Esempio (da base 10 a base 16)

Dividere 317_{10} per 16_{10}
(in base 16, le cifre sono 0,1,…,9,A,B,…,F)

1) 317 **:** 16

$Q=19_{10}$, $r=13_{10}$

$13_{10}=\mathbf{D}_{16}$ (LSD)

2) 19 **:** 16

$Q=1_{10}$, $r=3_{10}$

$3_{10}=\mathbf{3}_{16}$

3) 1 **:** 16

$Q=0_{10}$, $r=1_{10}$

$1_{10}=\mathbf{1}_{16}$ (MSD)

Quindi

$317_{10}=13D_{16}$

N.B.: nell'algoritmo sopra descritto, **la divisione va eseguita in base a (cioè nella base del numero di partenza)**. Se a≠10, questo è complicato.

Esempio (basi di partenza e arrivo diverse da 10)

Convertire 102202_3 in base 5

Tre strade :

a) eseguire ripetutamente 102202_3 **:** 12_3

N.B.: divisione con aritmetica in base 3 → **DIFFICILE!**

b) Applicare il metodo polinomiale a 102202_3

N.B.: prodotti e somme con aritmetica in base 5 → **DIFFICILE!**

c) convertire 102202_3 in base 10 (metodo polinomiale) e poi convertire il risultato in base 5 (divisioni iterate)

- $102202_3 = 3^5 + 2\cdot3^3 + 2\cdot3^2 + 2 = 317_{10}$
- 317:5 = 63 resto 2
  63:5 = 12 resto 3
  12:5 = 2 resto 2
  2:5 = 0 resto 2

Quindi, $102202_3 = 2232_5$

Conversioni da base *a* a base a^k

Prop.: nell'aritmetica in base *a* si ha che

$$c_{n-1} \dots c_1 c_0 \ \text{mod}\ a^k = c_{k-1} \dots c_0$$
$$c_{n-1} \dots c_1 c_0 \ \text{div}\ a^k = c_{n-1} \dots c_k$$

Esempio (a = 10 e k = 2): 453_{10} mod 100 = 53
453_{10} div 100 = 4 (essendo $100 = 10^2$)

Quindi, se $b = a^k$ e $N_a = c_{n-1} \dots c_1 c_0$, allora il numero in base b è

$$(c_{n-1} \dots c_{hk})_b \dots (c_{3k-1} \dots c_{2k})_b (c_{2k-1} \dots c_k)_b (c_{k-1} \dots c_0)_b$$

Caso specifico: *Conversione da base 2 a base 2^k*
Considera i bit a k-ple partendo dal meno significativo e traducile in base 2^k

Esempio: convertire 1000111101 da base 2 a base 4 (= 2^2)
$(10\ 00\ 11\ 11\ 01)_2 = (2\ 0\ 3\ 3\ 1)_4$

Esempio

Convertire 101001101101 da base 2 a base 8 e 16
OSS: $8=2^3$ e $16=2^4$

Prima conversione: dividi in triple e converti
$(101\ 001\ 101\ 101)_2 \rightarrow (5\ 1\ 5\ 5)_8$
Seconda conversione: dividi in quadruple e converti
$(1010\ 0110\ 1101)_2 \rightarrow (A\ 6\ D)_{16}$

OSS: nel dividere in gruppi da k bit, il gruppo più significativo potrebbe avere meno di k bit (se la stringa di partenza non ha lunghezza pari a un multiplo di k); in tal caso vanno aggiunti degli zeri

Es: 1001101101 da base 2 a base 8: 001 001 101 101

Conversioni da base b^k a base b

Similmente, se $a = b^k$ e $N_a = c_{n-1} \ldots c_1 c_0$, allora il numero in base b è

$$(c_{n-1})_b \ldots (c_2)_b\, (c_1)_b\, (c_0)_b$$

dove ogni c_i è convertita in base b usando k cifre.

Esempio 1: Convertire 8315_9 in base 3

Essendo $9 = 3^2$, converto ogni cifra del numero dato in base 9 con due cifre in base 3:

$$8\ 3\ 1\ 5_{\ 9} = 22\ 10\ 01\ 12_{\ 3}$$

Esempio 2: Convertire $8D3A_{16}$ in base 2

Essendo $16 = 2^4$, converto ogni cifra del numero dato in base 16 con quattro bit:

$$8\ D\ 3\ A_{\ 16} = 1000\ 1101\ 0011\ 1010_{\ 2}$$

Per convincervi dell'esattezza del metodo, fate le conversioni inverse come spiegato nei lucidi precedenti!

Conversione di base: Riassunto

- Il metodo polinomiale è facile se la base di arrivo è 10
- Il metodo delle divisioni iterate è facile se la base di partenza è 10
- Se né la base di partenza né quella di arrivo è 10:

  Soluzione generale:
  - Converto N_a in base 10 col metodo polinomiale
  - Converto il numero ottenuto in base b col metodo delle divisioni iterate

  Se la base di arrivo è potenza della base di partenza ($b = a^k$):
  - Converto in base b k-ple di cifre, dalla meno alla più significativa, di N_a
  - Le cifre così ottenute danno le cifre, dalla meno alla più significativa, del numero rappresentato in base b

  Se la base di partenza è potenza della base di arrivo ($a = b^k$):
  - Converto in base b ogni cifra di N_a usando k cifre (di base b)

Numero di sequenze binarie

Avendo a disposizione n bit, quante diverse sequenze posso creare?

1 bit:	0	,	1	→ 2 sequenze
2 bit:	00,01	,	10,11	→ 4 sequenze
3 bit:	000,001,010,011,		100,101,110,111	→ 8 sequenze
4 bit:	…		…	→ 16 sequenze

…

Ad ogni passo raddoppio le sequenze del passo precedente

n bit: $\underbrace{2 \cdot 2 \cdot 2 \ldots \cdot 2}_{n} = 2^n$ sequenze

Intervallo di Rappresentazione

Quindi, con n bit, rappresentiamo 2^n numeri: $\{0,\ldots,2^n\text{-}1\}$

Infatti, il numero più piccolo che possiamo rappresentare è

$$\underbrace{0 \ldots 0}_{n} = \underbrace{0 + \ldots + 0}_{n} = 0$$

mentre il numero più grande che possiamo rappresentare è

$$\underbrace{1 \ldots 1}_{n} = \sum_{i=0}^{n-1} 2^i = 2^n - 1 \qquad \text{(è la serie geometrica!)}$$

Lunghezza della rappresentazione

Di quanti bit abbiamo bisogno per rappresentare un numero N_2?

Risposta: dobbiamo trovare il più piccolo $n \in \mathbf{N}$ tale che

$$N < 2^n$$

OSS: $\log_b k$ è l'esponente che bisogna dare alla base (b) per ottenere l'argomento (k), cioè

$$k = b^{\log_b k}$$

Quindi, ponendo $k = N$ e $b = 2$, la soluzione di $N = 2^n$ è $n = \log_2 N$

N.B.: in generale, n è un numero irrazionale. Siccome voglio un naturale e non voglio l'uguaglianza esatta, prendo $\lfloor \log_2 N \rfloor + 1$

Esempi: $N = 57$: $\log_2 N = 5{,}8328\ldots$

quindi ho bisogno di 5+1 bit (infatti $57_{10} = 111001_2$)

$N = 64$: $\log_2 N = 6$

quindi ho bisogno di 6+1 bit (infatti $64_{10} = 1000000_2$)

Lunghezza di parole di codice

Per semplicità, gli elaboratori lavorano su parole di codice di lunghezza fissata, tipicamente, potenze di 2 bit:

- 8 bit = *byte*
- 16 bit = *half-word*
- 32 bit = *word*
- 64 bit = *long word*

Sia k la lunghezza della parola di codice adottata:

- se un numero è rappresentabile con esattamente k bit, siamo a posto
- se un numero è rappresentabile con meno di k bit (diciamo n), dobbiamo inserire in testa k-n zeri (non significativi)
- se un numero, per essere rappresentato, richiede più di k bit?

  Si considerano solo i k bit meno significativi del numero

  → situazione di errore detta *overflow*