

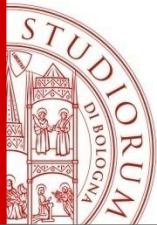
Seminari di Preparazione alle prove delle Olimpiadi di Informatica

Luciano Bononi

(bononi@cs.unibo.it)

Dipartimento di Scienze dell'Informazione, Università of Bologna
(in futuro Dipartimento di Informatica: Scienza e Ingegneria)

Bologna – 06-07/10/2011



Presentazione del Docente di oggi

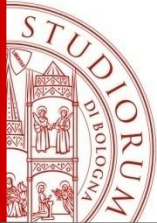
Luciano Bononi

- **Professore Associato presso il Dipartimento di Scienze dell'Informazione**

- **docente titolare dei corsi universitari di:**
 - Algoritmi e Strutture Dati (Laurea Triennale in Informatica per il Management)
 - Sistemi e Reti Wireless (Lauree Magistrali in Scienze di Internet e Informatica)
 - Laboratorio di Applicazioni Mobili (Laurea Triennale in Informatica per il Management)
 - Algorithms and Data Structures (International Master in Bioinformatics)

- **Ricerca e sviluppo:**
 - Progettazione e analisi di protocolli e architetture di rete mobile e wireless
 - Reti di comunicazione wireless veicolari
 - Sistemi di calcolo e comunicazione pervasivi
 - Reti su Chip
 - Sistemi cooperativi
 - “Smart Environments” e “Smart objects”
 - Internet of Things e applicazioni mobili

Per saperne di più: <http://www.cs.unibo.it/~bononi/>



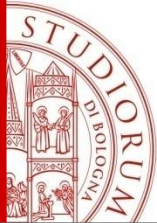
Scopo dei seminari

12-ma edizione Olimpiadi di Informatica

- Scopo: fare emergere e valorizzare le eccellenze
- Scuole dell'Emilia Romagna hanno conseguito grandi risultati

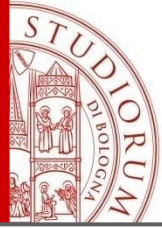
L' Ufficio Scolastico Regionale dell'Emilia Romagna e il Dipartimento di Scienze dell'Informazione dell'Università di Bologna organizzano questo primo ciclo di **seminari di preparazione** alle prove della fase scolastica

- Seguiranno seminari su temi e metodologie più approfondite in preparazione alle selezioni successive territoriali, regionali, nazionali, ecc.
- Grande contributo organizzativo e di preparazione dei docenti delle Scuole Superiori
- Seguiranno iniziative simili in merito alle Olimpiadi di Problem Solving
- **Per cominciare:** presenteremo esempi di metodologie e soluzioni di problemi classici dell'Informatica, a partire da semplici problemi di problem solving, fino a presentare applicazioni specifiche a problemi simili a quelli proposti nell'ambito delle prove olimpiche.



Alcuni temi dei seminari di questa fase

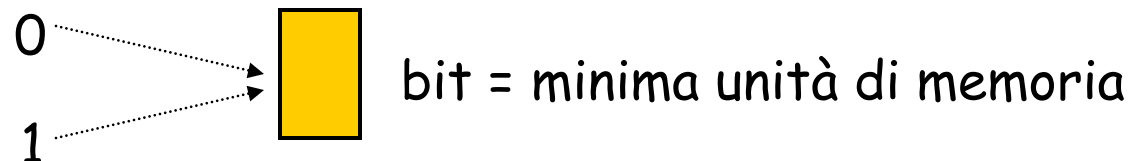
- Contare, Enumerare, Ottimizzare (un approccio che parte dalla logica e dal problem solving)
- Allenarsi a pensare in “modo algoritmico”
- Sistemi di numerazione (Rappresentazione posizionale di numeri e calcoli in basi diverse da 10)
- Esecuzione manuale di codice non ricorsivo
 - Tecniche di tracing
 - Tecniche di debugging
- Esecuzione manuale di codice ricorsivo
- Riconoscimento del problema risolto da una porzione di codice
- **Di questi, oggi vediamo:**
 - Sistemi di numerazione binaria (base due), una generalizzazione del concetto a basi qualsiasi, e una insospettata applicazione a un problema “trasversale”
 - Un paio di problemi di logica
 - Cenni sulle tecniche ricorsive di soluzioni a un problema (con esempi di esecuzione manuale di codice ricorsivo).



Numeri interi in rappresentazione binaria

La rappresentazione dei dati sul calcolatore

- **il calcolatore dispone di soli due simboli per la memorizzazione, rappresentazione e trasmissione di dati**
 - zero e uno (0,1) = valori assumibili da un binary digit (bit)
 - ogni valore deve essere rappresentato usando questi due simboli (notazione binaria)
 - esistono infinite interpretazioni (codifiche) che possono associare sequenze generiche di simboli 0 e 1 a valori numerici
 - noi vedremo alcune delle più significative



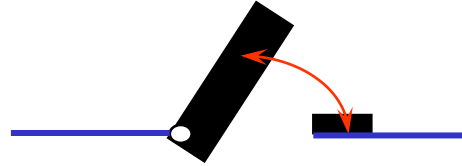
Il bit: valore e logica positiva (lp) e negativa (ln)



levetta:

alta = 1 (lp) 0(ln)

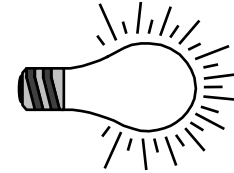
bassa = 0 (lp) 1(ln)



interruttore:

aperto = 0 (lp) 1 (ln)

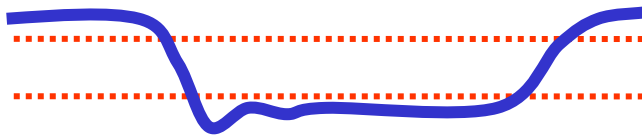
chiuso = 1 (lp) 0 (ln)



lampadina:

spenta = 0 (lp) 1 (ln)

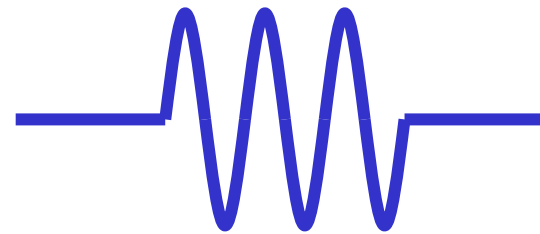
accesa = 1 (lp) 0 (ln)



tensione elettrica:

bassa = 0 (lp) 1 (ln)

alta = 1 (lp) 0 (ln)



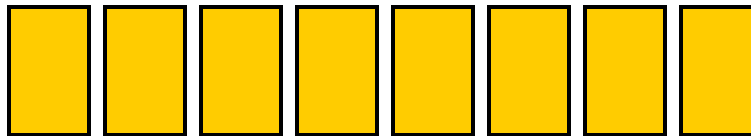
corrente elettrica 50 Hz:

assente = 0 (lp) 1 (ln)

presente = 1 (lp) 0 (ln)

La rappresentazione dei dati sul calcolatore

- i bit possono essere considerati in sequenza (in memoria)
 - sequenza di 8 bit = 1 Byte



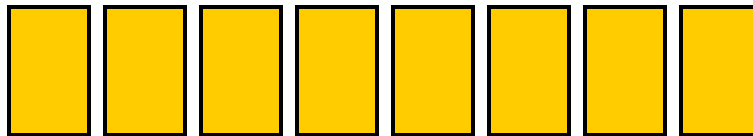
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	1
⋮							
1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1

2^n possibili sequenze diverse da n bit

n = numero di bit considerati

La rappresentazione dei dati sul calcolatore

- Dato un byte come possiamo associarvi i simboli o valori?
 - sequenza di 8 bit = 1 Byte (primo esempio poco utile)



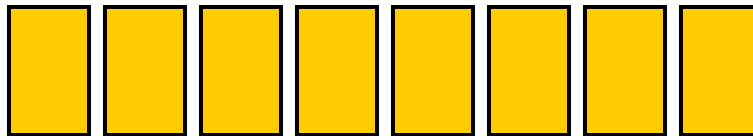
0	0	0	0	0	0	0	1	= valore 1
0	0	0	0	0	0	1	0	= valore 2
0	0	0	0	0	1	0	0	= valore 3
0	0	0	0	1	0	0	0	= valore 4
0	0	0	1	0	0	0	0	= valore 5
0	0	1	0	0	0	0	0	= valore 6
0	1	0	0	0	0	0	0	= valore 7
1	0	0	0	0	0	0	0	= valore 8

Solo valori da 1 a 8 ?
non sfrutto tutte le
possibili combinazioni!!!



La rappresentazione dei dati sul calcolatore

- Dato un byte come possiamo associarvi i simboli o valori?
 - sequenza di 8 bit = 1 Byte



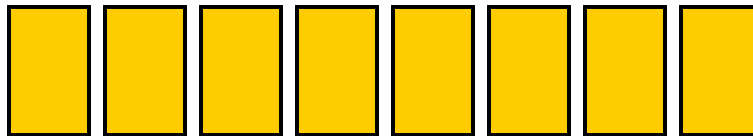
0	0	0	0	0	0	0	0	= valore 0
0	0	0	0	0	0	0	1	= valore 1
0	0	0	0	0	0	1	0	= valore 2
0	0	0	0	0	0	1	1	= valore 3
0	0	0	0	0	1	0	0	= valore 4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
1	1	1	1	1	1	1	0	= valore 254
1	1	1	1	1	1	1	1	= valore 255

Idea: sfruttare tutte le possibili combinazioni diverse di 8 bit
256 valori [0..255]



La rappresentazione dei dati sul calcolatore

- Dato un byte come possiamo associarvi i simboli o valori?
 - sequenza di 8 bit = 1 Byte



Idea: sfruttare tutte le possibili combinazioni diverse di 8 bit
256 simboli

0	0	0	0	0	0	0	0	= simbolo A
0	0	0	0	0	0	0	1	= simbolo B
0	0	0	0	0	0	1	0	= simbolo C
0	0	0	0	0	0	1	1	= simbolo E
0	0	0	0	0	1	0	0	= simbolo F
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
1	1	1	1	1	1	1	0	= simbolo %
1	1	1	1	1	1	1	1	= simbolo \$

Interpretazione (funzione valore)

- Nei sistemi posizionali, i simboli di una configurazione possono essere interpretati come i coefficienti del seguente polinomio [1] (detto funzione Valore)

$$V = \sum_{i=-m}^{n-1} d_i \cdot B^i$$

B = base

d_i = i-esima cifra $\in [0..B-1]$

n = numero di cifre parte intera

m = numero di cifre parte frazionaria

La virgola e' posta tra le cifre di posizione 0 e -1.

Interpretazione (funzione valore)

Esempio: sistema decimale

Il numero **245.6** decimale può essere rappresentato come segue:

$B = 10$

base

$n=3$

numero cifre parte intera

$m=1$

numero cifre parte frazionaria

$d = \{2,4,5,6\}$

insieme delle cifre

cifra	2	4	5	6
posizione	2	1	0	-1
peso	10^2	10^1	10^0	10^{-1}

$$V = \sum_{i=-m}^{n-1} d_i \cdot B^i = 2 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 + 6 \cdot 10^{-1} = 245.6$$

Esempio di interpretazione valore binario naturale

Esempio: Quale è il valore decimale corrispondente al numero binario 1101.010_2 ?

cifra ₂	1	1	0	1	.	0	1	0...
peso	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
valore	$1 \cdot 8$	$1 \cdot 4$	$0 \cdot 2$	$1 \cdot 1$.	$0 \cdot 1/2$	$1 \cdot 1/4$	$0 \cdot 1/8$

$$1101.010_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^0 + 1 \cdot 2^0 = 13.25_{10}$$

Algoritmo della divisione: da Base 10 a Base B

Ma come si può passare in modo pratico dalla rappresentazione di un valore in base 10 a un valore in qualsiasi base B (incluso $B=2$ per la notazione binaria)?

Algoritmo della divisione!

Metodo della divisione: Base 10 -> binario naturale

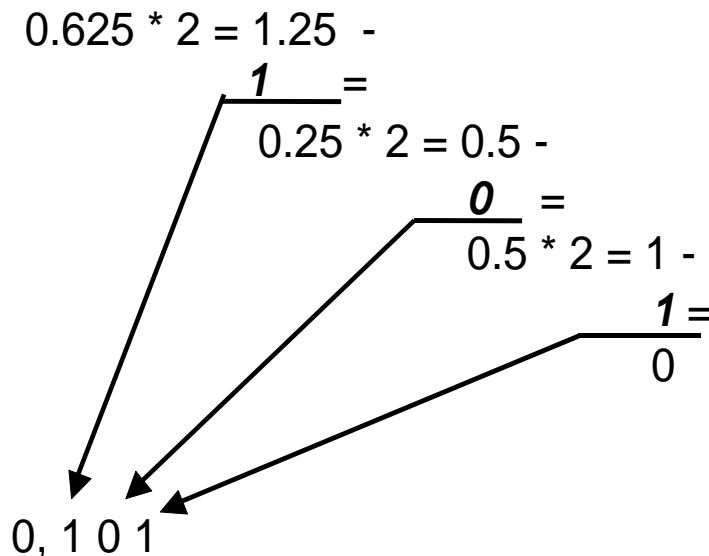
Es. voglio ottenere la parte decimale in base $B=2$

Per valori con cifre decimali: si separa la parte intera da quella frazionaria, La parte intera si calcola come nel caso precedente La parte frazionaria si ottiene come segue:

1. Si moltiplica la parte frazionaria per 2 il valore della base (2)
2. Se la parte intera del numero ottenuto è maggiore di 1, si sottrae il valore della parte intera (1), altrimenti si sottrae 0 e si prosegue
3. Si ripete dal passo 1 fino a che il numero ottenuto dopo la sottrazione è zero, oppure quando sono esaurite le cifre a disposizione

La sequenza dei valori sottratti ad ogni passo è la rappresentazione decimale ottenuta

$$\text{Esempio: } 0.625_{10} = 0.101_2$$



Somma di valori in binario naturale

Assumiamo di avere solo $n=3$ bit a disposizione per rappresentare i valori e il risultato, quindi posso rappresentare $B^n = 8$ valori diversi interi positivi (da 0 a 7).

$$5_{10} + 1_{10} = 6_{10}$$

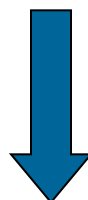
$$\begin{array}{r} 1 \\ 101 + \\ 001 = \\ \hline 110 \end{array}$$

$$2_{10} + 3_{10} = 5_{10}$$

$$\begin{array}{r} 1 \\ 010 + \\ 011 = \\ \hline 101 \end{array}$$

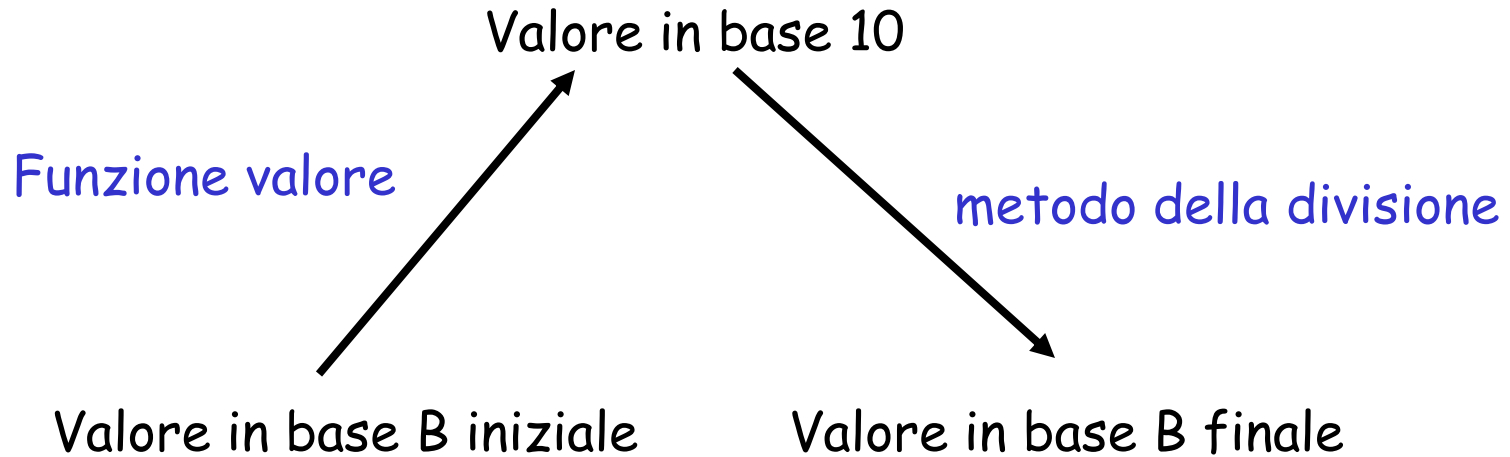
$$5_{10} + 3_{10} = 8_{10}$$

$$\begin{array}{r} 111 \\ 101 + \\ 011 = \\ \hline 1000 \end{array}$$


Overflow!!

Per rappresentare il risultato
della somma di $5_{10} + 3_{10}$ sono necessari 4 bit !

Riassunto: passaggi di base generici



Considerazioni utili

Data una base B , si possono esprimere tutti i valori naturali da da 0 a $(B^n)-1$

Infatti, con la tecnica posizionale (binaria, $B=2$) si possono esprimere tutti i valori naturali da 0 a $(2^n)-1$

Come vedremo, tale tecnica permette di enumerare anche tutti i casi possibili delle combinazioni di valori (vero o falso) di una serie di n predicati logici (eventi) indipendenti.

Tutte queste considerazioni sono utili alla soluzione di problemi (non solo informatici)

Esempio veloce: un problema già visto....

Problema: Mario ha 83 macchinine che deve riporre in sacchetti. La suddivisione in sacchetti deve essere fatta in modo tale che quando i compagni di gioco di Mario gli chiederanno un numero qualsiasi di macchinine (compreso fra 1 e 83), Mario sarà in grado di consegnare il numero giusto di macchinine porgendo un certo numero di sacchetti senza aprirli per modificarne il contenuto. Quale è il numero minimo di sacchetti che Mario deve usare per riporre le sue macchinine?

Pensiamo ai sacchetti S_n riempiti in questo modo (es. s_6 contiene 32 oggetti), e ci fermiamo quando n è tale che $2^{(n+1)}$ supera il valore di macchinine (83).

s_7	s_6	s_5	s_4	s_3	s_2	s_1
?	32	16	8	4	2	1

Idea dell'algoritmo: possiamo esprimere qualsiasi valore da 1 a 63 usando una delle combinazioni possibili di sacchetti determinate con il metodo della divisione.

Es. $43 / 2 = 21 / 2 = 10 / 2 = 5 / 2 = 2 / 2 = 1 / 2 = 0 / 2 = 0 / 2 \dots\dots$

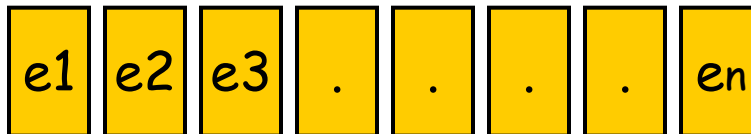
Resto 1 1 0 1 0 1 0..... da qui in poi sono tutti zero...

Per cui 43 in binario si scrive0101011. Prendere i sacchetti ai quali corrisponde il valore 1 significa prendere i sacchetti $s_1, s_2, s_4, s_6 = 1 + 2 + 8 + 32 = 43$ macchinine!

Se il sacchetto s_7 contiene il valore di macchinine totali (83) meno il valore massimo rappresentabile da s_1-s_6 (63), cioè $s_7=20$, allora posso esprimere tutti i valori X da 63 a 83 fornendo il sacchetto s_7 + la differenza $X-63$ espressa con il metodo della divisione applicato ai sacchetti rimanenti. **Quindi il numero minimo di sacchetti è 7!**

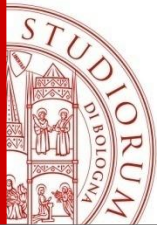
Enumerazione binaria dei casi vero o falso di una serie di eventi (o predicati logici)

- N.B. un'applicazione del concetto di enumerazione binaria si applica per avere un metodo semplice di enumerazione di tutti i possibili casi di combinazioni dei valori Vero o Falso di una serie di eventi indipendenti. Ciò è utile in molte occasioni.



0	0	0	0	0	0	0	0	= tutti gli eventi sono falsi
0	0	0	0	0	0	0	1	= solo l'evento e_n è vero
0	0	0	0	0	0	1	0	= solo l'evento e_{n-1} è vero
0	0	0	0	0	0	1	1	= solo gli eventi e_n e e_{n-1} sono veri
0	0	0	0	0	1	0	0	= solo l'evento e_{n-2} è vero
:	:	:	:	:	:	:	:	
1	1	1	1	1	1	1	0	= solo l'evento e_n è falso
1	1	1	1	1	1	1	1	= tutti gli eventi sono veri

Idea: se 0=falso, 1=vero quante possibili combinazioni diverse di n eventi sono possibili? 2^n



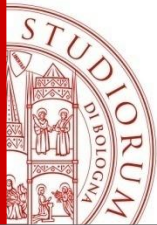
Un problema di logica

- **Il principio di non contraddizione (Aristotele, Metafisica) dice, intuitivamente, che “Nessuno può ritenere che la medesima cosa sia e non sia...”**
- Ciò può essere usato come argomento per la dimostrazione di ipotesi a partire da un dominio di enunciati e possibilità.
- Vediamo un esempio: *“Nel Mercante di Venezia di Shakespeare, Porzia aveva tre scrigni, uno d’oro, uno d’argento e uno di piombo e in uno solo c’era un suo ritratto. Porzia voleva scegliere il suo sposo sulla base della sua intelligenza e fece incidere le seguenti iscrizioni sugli scrigni:*
 - Scrigno d’oro: il ritratto e’ in questo scrigno (d’oro)*
 - Scrigno d’argento: il ritratto NON e’ in questo scrigno (d’argento)*
 - Scrigno di piombo: il ritratto NON e’ nello scrigno d’ORO*

*Porzia spiegò al suo pretendente che di queste tre affermazioni **al massimo una era vera.***

Q: Quale scrigno contiene il ritratto di Porzia?”

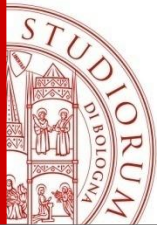
Affrontiamo nell’ordine le possibili ipotesi sulle veridicità delle frasi, eliminando tutte le possibilità.



Un problema di logica

Affrontiamo nell'ordine le possibili ipotesi sulle veridicità delle frasi, eliminando tutte le possibilità. Uso la tecnica di enumerazione binaria... sostituendo i simboli Vero e Falso a 1 e 0 per i tre casi.

Scrigno d'oro <i>"il ritratto e' in questo scrigno"</i>	Scrigno d'argento <i>"il ritratto NON e' in questo scrigno "</i>	Scrigno di piombo <i>"il ritratto NON e' nello scrigno d'ORO "</i>
F	F	F
F	F	V
F	V	F
F	V	V
V	F	F
V	F	V
V	V	F
V	V	V



Un problema di logica

Ora però devo eliminare i casi esclusi dalla condizione affermata nelle ipotesi del problema secondo cui
“*di queste tre affermazioni **al massimo una è vera***”.

Scrigno d'oro “il ritratto e' in questo scrigno”	Scrigno d'argento “il ritratto NON e' in questo scrigno ”	Scrigno di piombo “il ritratto NON e' nello scrigno d'ORO ”	
F	F	F	
F	F	V	
F	V	F	
F	V	V	(2 vere)
V	F	F	
V	F	V	(2 vere)
V	V	F	(2 vere)
V	V	V	(3 vere)

Un problema di logica

Per cui rimangono le seguenti ipotesi da verificare una a una.

1) Vediamo la prima riga (sono tutte false):

Qui dice che è falso che il ritratto sia nello scrigno d'oro

Qui dice che è falso che il ritratto NON sia nello scrigno d'oro

Ciò non è possibile per il principio di non contraddizione, quindi questa ipotesi è da scartare.

Scrigno d'oro "il ritratto e' in questo scrigno"	Scrigno d'argento "il ritratto NON e' in questo scrigno "	Scrigno di piombo "il ritratto NON e' nello scrigno d'ORO "	
F	F	F	Assurdo
F	F	V	
F	V	F	
V	F	F	

Un problema di logica

2) Vediamo la terza riga (è vera la scritta sullo scrigno d'argento):

Qui dice che è vero che il ritratto non è nello scrigno d'argento (quindi o è in quello d'oro o in quello di piombo)

ma allora, siccome una sola affermazione è vera, qui dice che non è nello scrigno d'oro (quindi può essere solo in quello di piombo... ma allora deve essere vera la scritta sullo scrigno di piombo.

Ciò non è possibile per il principio di non contraddizione, quindi questa ipotesi è da scartare.

Scrigno d'oro "il ritratto e' in questo scrigno"	Scrigno d'argento "il ritratto NON e' in questo scrigno "	Scrigno di piombo "il ritratto NON e' nello scrigno d'ORO "	
F	F	F	Assurdo
F	F	V	
F	V	F	Assurdo
V	F	F	

Un problema di logica

3) Vediamo la quarta riga (è vera la scritta sullo scrigno d'oro):

Qui dice che è vero che il ritratto non è nello scrigno d'oro (quindi vale anche la falsità della scritta su quello di piombo... ok!)

come può essere falso che il ritratto non è nello scrigno d'argento?
(ovvero è anche nello scrigno d'argento?)

Ciò non è possibile per il principio di non contraddizione, quindi questa ipotesi è da scartare.

	Scrigno d'oro "il ritratto e' in questo scrigno"	Scrigno d'argento "Il ritratto NON e' in questo scrigno "	Scrigno di piombo "il ritratto NON e' nello scrigno d'ORO "	
	F	F	F	Assurdo
	F	F	V	
	F	V	F	Assurdo
	V	F	F	Assurdo

Un problema di logica

4) Vediamo la seconda riga (è vera la scritta sullo scrigno di piombo):
 Qui esclude il fatto che il ritratto sia nello scrigno d'oro (confermato anche dalla falsità della scritta sullo scrigno d'oro).

Ma la scritta sullo scrigno d'argento è falsa, e quindi il ritratto DEVE trovarsi nello scrigno d'argento!

Il ritratto è nello scrigno d'argento!

Scrigno d'oro "il ritratto e' in questo scrigno"	Scrigno d'argento "il ritratto NON e' in questo scrigno"	Scrigno di piombo "il ritratto NON e' nello scrigno d'ORO"	
F	F	F	Assurdo
F	F	V	OK: argento!
F	V	F	Assurdo
V	F	F	Assurdo

Un secondo problema di logica

Date le affermazioni (entrambe vere):

1) tutti i belli (B) sono ricchi (R)

2) tutti i ricchi (R) sono tristi (T)

Quali delle seguenti affermazioni sono vere di conseguenza?

a) *alcuni tristi sono belli*

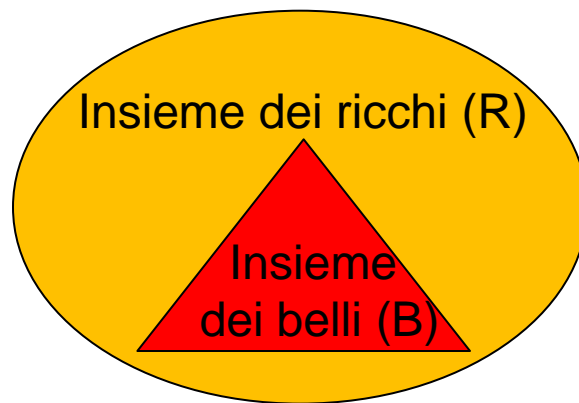
b) *tutti i ricchi sono belli*

c) *alcuni belli non sono tristi*

d) *nessuna delle precedenti*

Per questo problema usiamo una rappresentazione insiemistica, che aiuta a determinare il principio di inclusione e non contraddizione. Perché?

Cominciamo a considerare l'insieme dei Belli (B).... Per la prima affermazione, tutti i Belli sono anche Ricchi.



Un secondo problema di logica

Date le affermazioni (entrambe vere):

1) tutti i belli (B) sono ricchi (R)

2) tutti i ricchi (R) sono tristi (T)

Quali delle seguenti affermazioni sono vere di conseguenza?

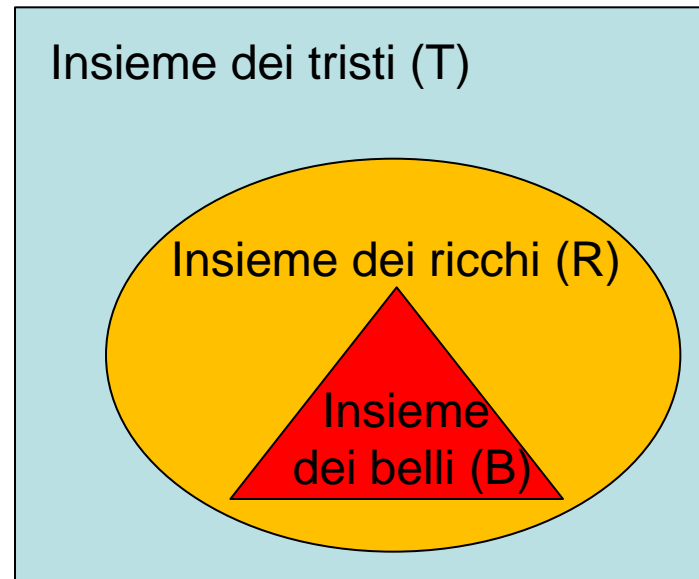
a) *alcuni tristi sono belli*

b) *tutti i ricchi sono belli*

c) *alcuni belli non sono tristi*

d) *nessuna delle precedenti*

Per la seconda affermazione, tutti i Ricchi sono anche Tristi...



Un secondo problema di logica

Date le affermazioni (entrambe vere):

1) tutti i belli (B) sono ricchi (R)

2) tutti i ricchi (R) sono tristi (T)

Quali delle seguenti affermazioni sono vere di conseguenza?

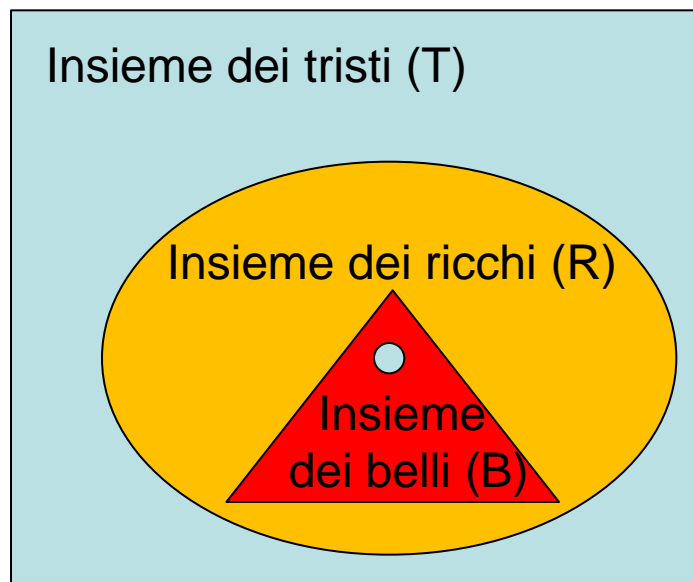
a) **alcuni tristi sono belli (vero)**

b) *tutti i ricchi sono belli*

c) *alcuni belli non sono tristi*

d) *nessuna delle precedenti*

Ora, supponendo che l'insieme B non sia vuoto, esiste almeno un bello che è anche triste. Quindi la a) è vera!



Un secondo problema di logica

Date le affermazioni (entrambe vere):

1) tutti i belli (B) sono ricchi (R)

2) tutti i ricchi (R) sono tristi (T)

Quali delle seguenti affermazioni sono vere di conseguenza?

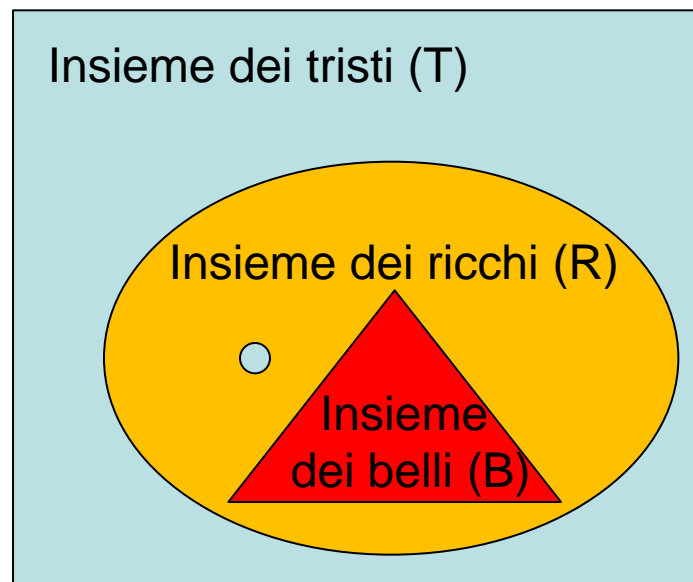
a) **alcuni tristi sono belli (vero)**

b) **tutti i ricchi sono belli (falso)**

c) *alcuni belli non sono tristi*

d) *nessuna delle precedenti*

Ora, supponendo che l'insieme R non sia vuoto, le affermazioni non escludono che esista almeno un ricco che non è bello. Quindi la b) è falsa!



Un secondo problema di logica

Date le affermazioni (entrambe vere):

1) tutti i belli (B) sono ricchi (R)

2) tutti i ricchi (R) sono tristi (T)

Quali delle seguenti affermazioni sono vere di conseguenza?

a) **alcuni tristi sono belli (vero)**

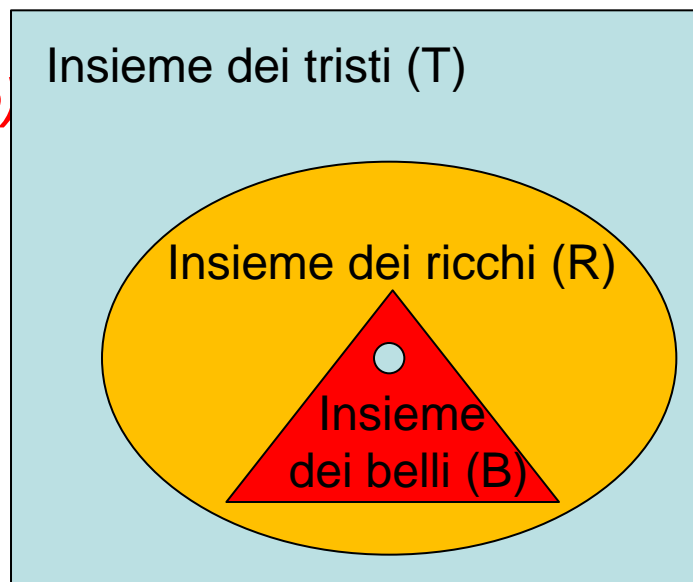
b) **tutti i ricchi sono belli (falso)**

c) **alcuni belli non sono tristi (falso)**

d) *nessuna delle precedenti*

Ora, supponendo che l'insieme B non sia vuoto, non è possibile che un bello, se esiste, non sia anche triste.

Quindi la c) è falsa!



Un secondo problema di logica

Date le affermazioni (entrambe vere):

1) tutti i belli (B) sono ricchi (R)

2) tutti i ricchi (R) sono tristi (T)

Quali delle seguenti affermazioni sono vere di conseguenza?

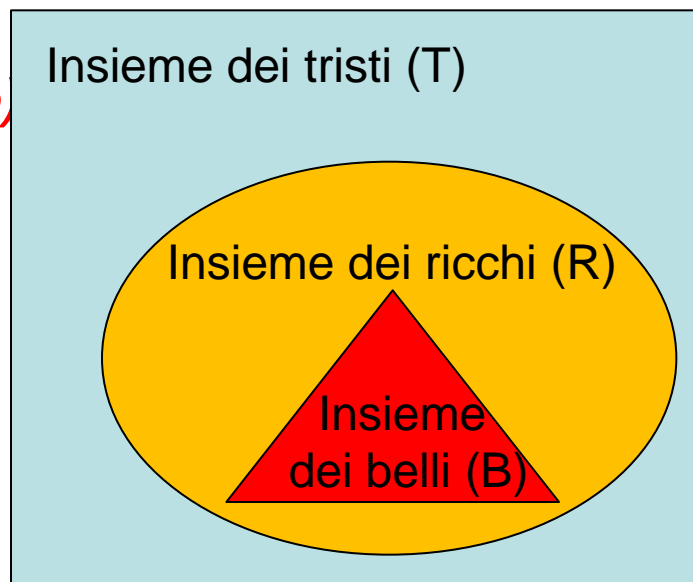
a) **alcuni tristi sono belli (vero)**

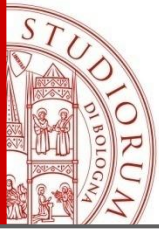
b) **tutti i ricchi sono belli (falso)**

c) **alcuni belli non sono tristi (falso)**

d) **nessuna delle precedenti (falso)**

Infine, essendo vera la a), la d) è falsa (per il principio di non contraddizione!)





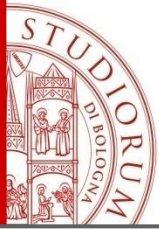
La ricorsione

Cambiamo genere....

Come si interpreta il significato di un programma ricorsivo? (ovvero il cui codice contiene di nuovo una chiamata (invocazione) dell'esecuzione del codice stesso.

Esempio:

```
int F(int N) {  
  if (N > 0)  
    return F(N - 1) + F(N - 1);  
  else  
    return 1;  
}
```



La ricorsione

Come viene eseguito un programma ricorsivo?

Ad esempio, eseguo F(3)!

Esecuzione di F(3):

```
{  
if (3 > 0)  
return F(2) + F(2);  
else  
return 1;  
}
```

Esecuzione di F(2):

```
{  
if (2 > 0)  
return F(1) + F(1);  
else  
return 1;  
}
```

```
int F(int N) {  
if (N > 0)  
return F(N - 1) + F(N - 1);  
else  
return 1;  
}
```

La ricorsione

Come viene eseguito un programma ricorsivo?

Ad esempio, eseguo F(3)!

```
int F(int N) {
  if (N > 0)
    return F(N - 1) + F(N - 1);
  else
    return 1;
}
```

Esecuzione di F(3):

```
{
  if (3 > 0)
    return F(2) + F(2);
  else
    return 1;
}
```

Ese

```
{
  if (2
  retu
  else
  retu
}
```

Esecuzione di F(2):

```
{
  if (2 > 0)
    return F(1) + F(1);
  else
    return 1;
}
```

Esecuzione di F(1):

```
E {
  E {
    E { if (1 > 0)
      { if return F(0) + F(0);
        { if re else
          { if re e return 1;
            re e re }
            e re }
            re }
            }
```


La ricorsione

Come viene eseguito un programma ricorsivo?
Ad esempio, eseguo F(3)!

```
int F(int N) {
  if (N > 0)
    return F(N - 1) + F(N - 1);
  else
    return 1;
}
```

Esecuzione di F(3):

```
{
  if (3 > 0)
    return F(2) + F(2);
  else
    return 1;
}
```

Ese

```
{
  if (2
  retu
  else
  retu
}
```

Esecuzione di F(2):

```
{
  if (2 > 0)
    return F(1) + F(1);
  else
    return 1;
}
```

Esecuzione di F(1):

```
E {
  if (1 > 0)
    return F(0) + F(0);
  else
    return 1;
}
E {
  if (1 > 0)
    return F(0) + F(0);
  else
    return 1;
}
E {
  if (1 > 0)
    return F(0) + F(0);
  else
    return 1;
}
E {
  if (1 > 0)
    return F(0) + F(0);
  else
    return 1;
}
```

Esecuzione di F(0):

Esecuzione di F(0):

Esecuzione di F(0):

Esecuzione di F(0):

Esecuzione di F(0):

```
{
  if (0 > 0)
    return F(-1) + F(-1);
  else
    return 1;
}
```


La ricorsione

Come viene eseguito un programma ricorsivo?
Ad esempio, eseguo F(3)!

```
int F(int N) {
  if (N > 0)
    return F(N - 1) + F(N - 1);
  else
    return 1;
}
```

Esecuzione di F(3):

```
{
  if (3 > 0)
    return F(2) + F(2);
  else
    return 1;
}
```

Ese

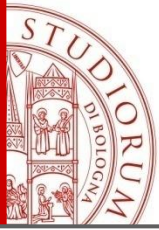
```
{
  if (2
  retu
  else
  retu
}
```

Esecuzione di F(2):

```
{
  if (2 > 0)
    return 2 + 2;
  else
    return 1;
}
```

Esecuzione di F(1):

```
E {
  E { if (1 > 0)
  E { if return 1 + 1;
    { if re else
    if re e return 1;
    re e re }
    e re }
    re }
  }
```



La ricorsione

Come viene eseguito un programma ricorsivo?

Ad esempio, eseguo F(3)!

Esecuzione di F(3):

```
{  
if (3 > 0)  
return 4 + 4;  
else  
return 1;  
}
```

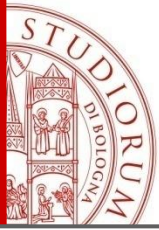
Ese

```
{  
if (2  
retu  
else  
retu  
}
```

Esecuzione di F(2):

```
{  
if (2 > 0)  
return 2 + 2;  
else  
return 1;  
}
```

```
int F(int N) {  
if (N > 0)  
return F(N - 1) + F(N - 1);  
else  
return 1;  
}
```



La ricorsione

Come viene eseguito un programma ricorsivo?

Ad esempio, eseguo F(3)!

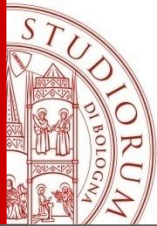
Esecuzione di F(3):

```
{  
if (3 > 0)  
return 4 + 4;  
else  
return 1;  
}
```

```
int F(int N) {  
if (N > 0)  
return F(N - 1) + F(N - 1);  
else  
return 1;  
}
```

Risultato: l'esecuzione di F(3) ritorna il valore 8 !

Quindi, ragionando per induzione al variare di $N=0$, $N=1$, $N=2$, ecc.
la funzione ritorna il valore 2^N



Conclusioni

Grazie per l'attenzione e in bocca al lupo per le vostre Olimpiadi.

Luciano Bononi (email: bononi@cs.unibo.it)

Trovate le prossime date di seminario su:

<http://www.cs.unibo.it/seminari-preparazione-olimpiadi-informatica/>

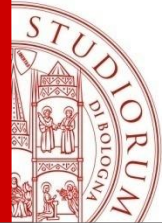
Prossime date:

Venerdì 7 Ottobre 2011, ore 15-17 (questa lezione);

Venerdì 14 Ottobre 2011, ore 15-17;

Venerdì 21 Ottobre 2011, ore 15-17;

Venerdì 28 Ottobre 2011, ore 15-17



Contatti per informazioni su Orientamento

- Dipartimento di Scienze dell'Informazione
 - Mura Anteo Zamboni 7, 40127, Bologna
- Orientamento
 - Web: <http://www.cs.unibo.it/orientamento/>
 - E-mail: orientamento.informatica@unibo.it
- Corso di Laurea in Informatica
 - <http://corsi.unibo.it/informatica/>
- Corso di Laurea Magistrale in Informatica
 - <http://corsi.unibo.it/informatica-magistrale/>
- Corso di Laurea in Informatica per il Management
 - <http://corsi.unibo.it/informaticamanagement/>
- Corso di Laurea Magistrale in Scienze di Internet
 - <http://corsi.unibo.it/magistrale/ScienzeInternet/>