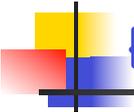




Teoria della Calcolabilità

- Si occupa delle questioni fondamentali circa la **potenza** e le **limitazioni** dei sistemi di calcolo.
- L'origine risale alla prima metà del ventesimo secolo, quando i logici matematici iniziarono ad esplorare i concetti di
 - computazione
 - algoritmo
 - problema risolvibile per via algoritmicae dimostrarono l'esistenza di **problemi non risolvibili**, ossia **problemi che non ammettono un algoritmo di risoluzione**.
⇒ **Problemi non decidibili**

1



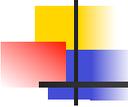
Problemi computazionali

Problemi formulati matematicamente di cui cerchiamo una soluzione algoritmica.

Classificazione:

- **problemi non decidibili**
- **problemi decidibili**
 - **problemi trattabili** (costo polinomiale)
 - **problemi intrattabili** (costo esponenziale)

2

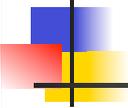


Calcolabilità e complessità

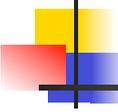
- **Calcolabilità:** nozioni di algoritmo e di problema non decidibile.
- **Complessità:** nozione di algoritmo efficiente e di problema intrattabile.
- La calcolabilità ha lo scopo di classificare i problemi in *risolvibili* e *non resolvibili*, mentre la complessità in "*facili*" e "*difficili*".

3

ESISTENZA DI PROBLEMI INDECIDIBILI



4



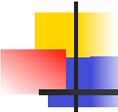
Insiemi numerabili

Un insieme è **numerabile** (possiede una infinità numerabile di elementi)

\longleftrightarrow

i suoi elementi possono essere messi in **corrispondenza biunivoca con i numeri naturali**.

5



Insiemi numerabili

- Un insieme numerabile è un insieme i cui elementi possono essere enumerati, ossia descritti da una sequenza del tipo

$$a_1, a_2, \dots, a_n, \dots$$

6

Insiemi numerabili: esempi

- Insieme dei numeri naturali \mathbb{N}
- Insieme dei numeri interi \mathbb{Z} :
 - $n \leftrightarrow 2n + 1 \quad n \geq 0$
 - $n \leftrightarrow 2|n| \quad n < 0$

$0, -1, 1, -2, 2, -3, 3, -4, 4, \dots$
- Insieme dei numeri naturali pari:
 - $2n \leftrightarrow n \quad 0, 2, 4, 6, 8, \dots$
- Insieme delle stringhe (sequenze) su un alfabeto finito.

7

Enumerazione delle sequenze

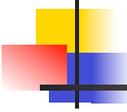
Si vogliono elencare in un ordine ragionevole le sequenze costruite su un alfabeto.

Le sequenze non sono in numero finito, quindi non si potrà completare l'elenco.

Scopo:

- raggiungere qualsiasi sequenza σ arbitrariamente scelta in un numero finito di passi
- σ deve dunque trovarsi a distanza finita dall'inizio dell'elenco.

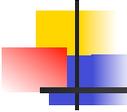
8



Ordinamento canonico

- *si ordinano le sequenze in ordine di lunghezza crescente
e, a pari lunghezza, in "ordine alfabetico";*
- *una sequenza s arbitraria si troverà tra quelle di $|s|$ caratteri, in posizione alfabetica tra queste.*

9



Esempio

$\Gamma = \{a, b, c, \dots, z\}$

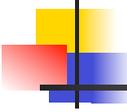
$a, b, c, \dots, z,$

$aa, ab, \dots, az, ba, \dots, bz, \dots, za, \dots, zz,$

$aaa, aab, \dots, baa, \dots, zaa, \dots, zzz,$

...

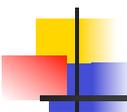
10



Enumerazione delle sequenze

- *ad una sequenza arbitraria corrisponde il numero che ne indica la posizione nell'elenco;*
- *a un numero naturale n corrisponde la sequenza in posizione n .*

11

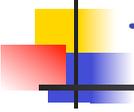


Insiemi non numerabili

Esempi:

- insieme dei numeri reali compresi nell'intervallo chiuso $[0,1]$
- insieme dei numeri reali compresi nell'intervallo aperto $(0,1)$
- insieme dei numeri reali
- insieme di tutte le linee nel piano
- insieme delle funzioni in una o più variabili.

12



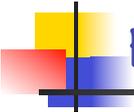
Teorema

L'insieme

$$F = \{ \text{funzioni } f \mid f: \mathbb{N} \rightarrow \{0,1\} \}$$

non è numerabile.

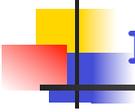
13



Funzioni

- $F = \{f: \mathbb{N} \rightarrow \{0,1\}\}$ non è numerabile
- A maggior ragione, non sono numerabili gli insiemi delle funzioni:
 - $f: \mathbb{N} \rightarrow \mathbb{N}$
 - $f: \mathbb{N} \rightarrow \mathbb{R}$
 - $f: \mathbb{R} \rightarrow \mathbb{R}$
 - $f: \mathbb{N}^k \rightarrow \mathbb{N}$
 - $f: \mathbb{N}^k \rightarrow \mathbb{N}^j$

14

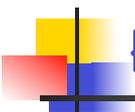


Il problema della rappresentazione

Drastica perdita di potenza:

*le rappresentazioni che possiamo costruire
con alfabeti finiti sono numerabili,
e sono meno delle funzioni matematiche.*

15



Problemi computazionali

L'insieme dei problemi computazionali
NON è numerabile.

16



Dalle funzioni ai problemi

Problema:

domanda posta su alcuni dati, e la sua soluzione consiste in una risposta, risultato, avente opportune proprietà.

17

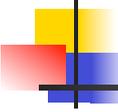


Dalle funzioni ai problemi

Dati e risultato sono rappresentati da sequenze di caratteri:

- *un problema specifica un'associazione tra sequenze (o interi che le rappresentano)*
- *la sua risoluzione corrisponde al calcolo di una funzione.*

18

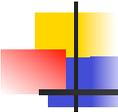


Problemi e funzioni

Un problema computazionale può essere visto come una **funzione matematica** che

- associa ad ogni insieme di dati**
- il corrispondente risultato**

19



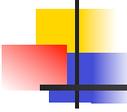
Dalle funzioni ai problemi

- *Esistono tanti problemi quante funzioni.*
- *Le funzioni non sono numerabili.*

⇒

- *I problemi non sono numerabili.*

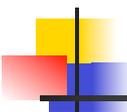
20



Il problema della rappresentazione

L'informatica rappresenta tutte le sue entità (quindi anche gli algoritmi) in forma digitale, come *sequenze finite di simboli di alfabeti finiti* (e.g., {0,1});
descrive dunque un *mondo numerabile*.

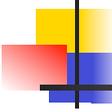
21



Il concetto di algoritmo

- Il concetto di **algoritmo** è l'elemento centrale della teoria della calcolabilità.
- Un **algoritmo** è un procedimento di calcolo che consente di pervenire alla soluzione di un problema, numerico o simbolico, mediante una *sequenza finita di operazioni, completamente e univocamente determinate*.

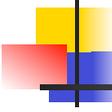
22



Caratteristiche essenziali

- l'**insieme di istruzioni** che definisce l'algoritmo è **finito**;
- l'**insieme delle informazioni** che rappresentano il problema e i **requisiti** richiesti alla sua soluzione hanno una descrizione **finita**;
- esiste un agente di calcolo in grado di eseguire le istruzioni;
- la computazione è **deterministica**: la sequenza dei passi computazionali è determinata senza alcuna ambiguità.

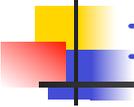
23



Algoritmi

- *La formulazione di un algoritmo dipende dal modello di calcolo utilizzato*
 - "programma" per un modello matematico astratto
 - programma in linguaggio Java per un PC, etc.
- *Qualsiasi modello si scelga, gli algoritmi devono esservi descritti, ossia rappresentati da sequenze finite di caratteri:*
 - ⇒ *gli algoritmi sono possibilmente infiniti, ma numerabili.*

24



Il problema della rappresentazione

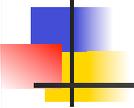
$|\{\text{Algoritmi}\}| \ll |\{\text{Problemi}\}|$

\Rightarrow

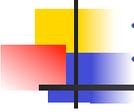
Esistono problemi per cui non esiste un algoritmo di calcolo!

25

UN PROBLEMA INDECIDIBILE



26



Il problema dell'arresto

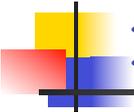
Abbiamo dimostrato l'esistenza di funzioni/
problemi non calcolabili.

I problemi che si presentano
spontaneamente sono tutti calcolabili.

Non è stato facile individuare un problema
che non lo fosse.

Turing (1930): **Problema dell'arresto.**

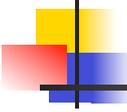
27



Il problema dell'arresto

- Considera algoritmi che indagano sulle proprietà di altri algoritmi, che sono trattati come dati.
- È legittimo: gli algoritmi sono rappresentabili con sequenze di simboli, che possono essere presi dallo stesso alfabeto usato per codificare i dati di input.
- Una stessa sequenza di simboli può essere quindi interpretata sia come un programma, sia come un dato di ingresso di un altro programma.

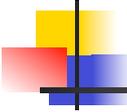
28



Il problema dell'arresto

- Un algoritmo A , comunque formulato, può operare sulla rappresentazione di un altro algoritmo B .
- Possiamo calcolare $A(B)$.
- In particolare può avere senso calcolare $A(A)$.

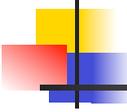
29



Il problema dell'arresto

*Presi ad arbitrio un **algoritmo** A e i suoi **dati di input** D , decidere in tempo finito se la computazione di A su D termina o no.*

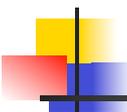
30



Il problema dell'arresto

Consiste nel chiedersi se un generico programma termina la sua esecuzione, oppure "va in ciclo", ovvero continua a ripetere la stessa sequenza di istruzioni all'infinito (supponendo di non avere limiti di tempo e memoria).

31



ESEMPIO:

Stabilire se un intero $p > 1$ è primo.

```
Primo(p)  
fattore = 2;  
while (p % fattore != 0)  
    fattore++;  
return (fattore == p);
```

Termina sicuramente (la guardia del **while** diventa falsa quando $\text{fattore} = p$).

32

ESEMPIO

- Programma che trova il più piccolo numero intero pari (maggiore di 4) che **NON** sia la somma di due numeri primi.
- Il programma si **arresta** quando trova $n \geq 4$ che **NON** è la somma di due primi.

33

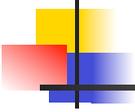
ESEMPIO

```

Goldbach()
n = 2;
do {
    n = n + 2;
    controesempio = true;
    for (p = 2; p ≤ n - 2; p++) {
        q = n - p;
        if (Primo(p) && Primo(q))
            controesempio = false;
    }
} while (!controesempio);
return n;

```

34



Congettura di Goldbach.

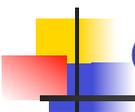
XVIII secolo

“ogni numero intero pari $n \geq 4$ è la somma di due numeri primi”

Congettura **falsa** → Goldbach() si arresta

Congettura **vera** → Goldbach() **NON** si arresta

35

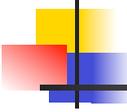


Osservazione

L'algorithm ARRESTO costituirebbe dunque uno strumento estremamente potente:

permetterebbe infatti di dimostrare congetture ancora aperte sugli interi (esempio: la congettura di Goldbach).

36



L'ultimo Teorema di Fermat

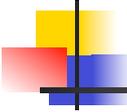
Per ogni a, b, c interi, e per ogni $n > 2$,

$$a^n + b^n \neq c^n$$

- Dimostrato nel 1994.
- Dimostrare il teorema equivale a negare l'esistenza di quattro valori interi per a, b, c, n tali che:

$$a^n + b^n = c^n$$

37

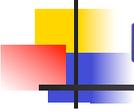


L'ultimo Teorema di Fermat

Si potrebbe scrivere un algoritmo TEST, che genera al suo interno tutte le infinite combinazioni di valori delle variabili, e verifica se $a^n + b^n = c^n$.

Non appena TEST trova quattro valori per cui $a^n + b^n = c^n$, l'algoritmo si arresta (in tempo finito, ma non noto).

38

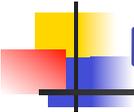


L'ultimo Teorema di Fermat

L'algoritmo TEST non aiuta a dimostrare il Teorema di Fermat:

Se il teorema fosse vero (e oggi sappiamo che lo è), TEST non si arresterebbe mai e la dimostrazione durerebbe in eterno.

39

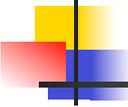


L'ultimo Teorema di Fermat

Se esistesse l'algoritmo ARRESTO, esso potrebbe essere utilizzato per indicare, in tempo finito, se la computazione di TEST si arresta o meno,

e quindi se l'ultimo Teorema di Fermat è falso o vero.

40



TEOREMA

Turing ha dimostrato che riuscire a dimostrare se un programma arbitrario si arresta e termina la sua esecuzione non è solo un'impresa ardua, ma in generale è IMPOSSIBILE!

TEOREMA
Il problema dell'arresto è INDECIDIBILE.

41



DIMOSTRAZIONE (per assurdo)

Se il problema dell'arresto fosse decidibile, allora esisterebbe un **algoritmo ARRESTO** che:

- presi A e D come dati di input
- determina in tempo finito le risposte:

ARRESTO(A,D) = 1 se A(D) termina
ARRESTO(A,D) = 0 se A(D) non termina

42



Osservazione

L'algoritmo ARRESTO non può consistere in un algoritmo che simuli la computazione $A(D)$:

se A non si arresta su D , ARRESTO non sarebbe in grado di rispondere NO (0) in tempo finito.

43



DIMOSTRAZIONE

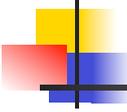
In particolare possiamo scegliere $D = A$,
cioè considerare la computazione $A(A)$:

$ARRESTO(A,A) = 1$

↔

$A(A)$ termina

44

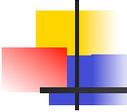


DIMOSTRAZIONE

Se esistesse l'algoritmo ARRESTO,
esisterebbe anche il seguente algoritmo:

```
PARADOSSO (A)  
while (ARRESTO(A,A)) {  
    ;  
}
```

45



DIMOSTRAZIONE

L'ispezione dell'algoritmo PARADOSSO
mostra che:

PARADOSSO(A) termina

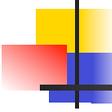


x = ARRESTO(A,A) = 0



A(A) non termina

46



DIMOSTRAZIONE

Cosa succede calcolando PARADOSSO(PARADOSSO)?

PARADOSSO(PARADOSSO) termina

↔

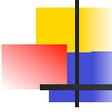
$x = \text{ARRESTO}(\text{PARADOSSO}, \text{PARADOSSO}) = 0$

↔

PARADOSSO(PARADOSSO) non termina

contraddizione!

47



DIMOSTRAZIONE

L'unico modo di risolvere la contraddizione è che l'algoritmo PARADOSSO non possa esistere.

Dunque non può esistere nemmeno l'algoritmo ARRESTO.

In conclusione, *il problema dell'arresto è indecidibile!*

48



Osservazione

*Non può esistere un algoritmo che decida in tempo finito se un algoritmo arbitrario A termina la sua computazione su dati arbitrari D .
ciò non significa che non si possa decidere in tempo finito la terminazione di algoritmi particolari:
il problema è indecidibile su una coppia $\langle A, D \rangle$ scelta arbitrariamente.*

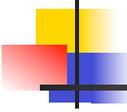
49



La "lezione" di Turing

Non possono esistere algoritmi che decidono il comportamento di altri algoritmi esaminandoli "dall'esterno", cioè senza passare dalla loro simulazione.

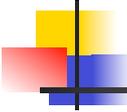
50



Il decimo problema di Hilbert

- *Esistono risultati di non calcolabilità relativi ad altre aree della matematica, tra cui la teoria dei numeri e l'algebra.*
- *Tra questi, occupa un posto di rilievo il ben noto **decimo problema di Hilbert.***

51



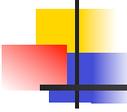
Equazioni diofantee

Un'equazione diofantea è un'equazione della forma

$$p(x_1, x_2, \dots, x_m) = 0$$

dove p è un polinomio a coefficienti interi.

52



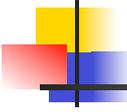
Il decimo problema di Hilbert

Data un'arbitraria equazione diofantea, di grado arbitrario e con un numero arbitrario di incognite

$$p(x_1, x_2, \dots, x_m) = 0$$

stabilire se p ammette soluzioni intere.

53



Teorema

Il decimo problema di Hilbert non è calcolabile.

54



Il decimo problema di Hilbert

La questione circa la calcolabilità di questo problema è rimasta aperta per moltissimi anni,

ha attratto l'attenzione di illustri matematici,

ed è stata risolta nel 1970 da un matematico russo allora poco più che ventenne, Yuri Matiyasevich.

55