



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 algoritmica
e dimostrarono l'esistenza di **problemi che non ammettono un algoritmo di risoluzione.**

⇒ ***Problemi non decidibili***



Problemi computazionali

Problemi formulati matematicamente di cui cerchiamo una soluzione algoritmica.

Classificazione:

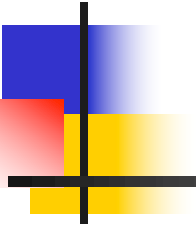
- **problemi non decidibili**
- **problemi decidibili**
 - **problemi trattabili** (costo polinomiale)
problemi presumibilmente intrattabili
 - **problemi intrattabili** (costo esponenziale)



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 risolvibili*, mentre la complessità in "*facili*" e "*difficili*"

ESISTENZA DI PROBLEMI INDECIDIBILI





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., $\{0,1\}$).



Il concetto di algoritmo

Algoritmo:

sequenza finita di operazioni, completamente e univocamente determinate



Algoritmi

La formulazione di un algoritmo dipende dal modello di calcolo utilizzato:

- *“programma” per un modello matematico astratto, come una Macchina di Turing*
- *algoritmo per in pseudocodice per RAM*
- *programma in linguaggio C per un PC*



Algoritmi

Qualsiasi modello si scelga, gli algoritmi devono esservi descritti,

ossia rappresentati da sequenze finite di caratteri di un alfabeto finito

*⇒ **gli algoritmi sono possibilmente infiniti, ma numerabili***

possono essere 'elencati' (messi in corrispondenza biunivoca con l'insieme dei numeri naturali)



Problemi computazionali

I problemi computazionali

(funzioni matematiche che associano ad ogni insieme di dati il corrispondente risultato)

non sono numerabili



Il problema della rappresentazione

Drastica perdita di potenza:

gli algoritmi sono numerabili,

e sono meno dei problemi computazionali, che hanno la potenza del continuo

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



Esistono problemi privi di un corrispondente algoritmo di calcolo!



Il problema dell'arresto

Esistono dunque 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**



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.*



Il problema dell'arresto

- Algoritmo che indaga sulle proprietà di un altro algoritmo, trattato come dato di input
- È legittimo: possiamo usare lo stesso alfabeto per codificare algoritmi e i loro dati di ingresso (sequenze di simboli dell'alfabeto)
- Una stessa sequenza di simboli può essere quindi interpretata sia come un programma, sia come un dato di ingresso di un altro programma



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 $A(A)$



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).



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 `fattore = p`).



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



Algoritmo GOLDBACH

- Scandisci in ordine crescente i numeri naturali pari maggiori di 2, fino a trovare un numero che **NON sia esprimibile come la somma di due numeri primi**
- Se e quando questo accade, stampa il numero e termina



Conggettura di Goldbach

XVIII secolo

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

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

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



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



DIMOSTRAZIONE

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



Osservazione

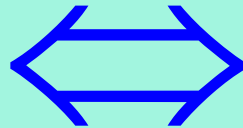
L'algoritmo ARRESTO non può consistere in un algoritmo che simuli la computazione $A(D)$

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

DIMOSTRAZIONE

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

$$\text{ARRESTO}(A,A) = 1$$



$A(A)$ termina



DIMOSTRAZIONE

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

PARADOSSO (A)

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




DIMOSTRAZIONE

L'ispezione dell'algoritmo PARADOSSO mostra che:

PARADOSSO(A) termina



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



A(A) non termina

DIMOSTRAZIONE

Cosa succede calcolando PARADOSSO(PARADOSSO)?

PARADOSSO(PARADOSSO) termina



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



PARADOSSO(PARADOSSO) non termina

contraddizione!



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 non è decidibile!*



Problemi indecidibili

- Altri problemi lo sono:
 - Ad esempio, è indecidibile stabilire l'**equivalenza** tra due programmi (se per ogni possibile input, producono lo stesso output)
- **“Lezione di Turing”**:

non esistono algoritmi che decidono il comportamento di altri algoritmi esaminandoli dall'esterno, cioè senza passare dalla loro simulazione.



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***



Equazioni diofantee

Un'equazione diofantea è un'equazione della forma

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

dove p è un polinomio a coefficienti interi.



Il decimo problema di Hilbert

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

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

stabilire se p ammette soluzioni intere.

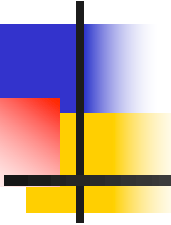


Teorema

Il decimo problema di Hilbert non è
calcolabile.

Yuri Matiyasevich, 1970

MODELLI DI CALCOLO E CALCOLABILITÀ





Modelli di calcolo

La teoria della calcolabilità dipende dal modello di calcolo?

oppure ...

la decidibilità è una proprietà del problema?