I limiti della computazione

Informatica e Bioinformatica – a.a 2023/24

Davide Bresolin
Ultimo aggiornamento: 16 ottobre 2024



Università degli Studi di Padova

Sommario



1 Problemi impossibili

Il metodo della diagonalizzazione

- Metodo scoperto da Cantor nel 1873
- Serve per confrontare le dimensioni di insiemi infiniti

Idea: due insiemi finiti hanno la stessa dimensione se gli elementi di un insieme possono essere accoppiati agli elementi dell'altro insieme.

Corrispondenze

- Abbiamo due insiemi A e B e una funzione $f : A \mapsto B$
- f è iniettiva se non mappa mai elementi diversi nello stesso punto: $f(a) \neq f(b)$ ogniqualvolta che $a \neq b$
- f è suriettiva se tocca ogni elemento di B: per ogni $b \in B$ esiste $a \in A$ tale che $f(a) = b$
- Una funzione iniettiva e suriettiva è chiamata biettiva: è un modo per accoppiare elementi di A con elementi di B

Definition

A e B hanno la stessa cardinalità se esiste una funzione biettiva $f : A \mapsto B$

Esempio: naturali vs numeri pari

Esempio

- $\mathbb{N} = \{0, 1, 2, \ldots, \}$, insieme dei numeri naturali
- $\mathbb{E} = \{0, 2, 4, \ldots, \}$, insieme dei numeri pari

Quale dei due insiemi è il più grande?

Definition (Insieme numerabile)

Un insieme è numerabile se è finito oppure ha la stessa cardinalità di $\mathbb{N}$

Esempio 2: $[0, 1]$ vs $[0, 2]$

Esempio 3: $\mathbb{N}$ vs $[0, 1]$

Esempio 4: $\mathbb{N}$ ed i programmi Python



Esempio 5: $\mathbb{N}$ e le funzioni $\mathbb{N} \mapsto \mathbb{N}$



Un risultato negativo

"Esistono funzioni che non sono algoritmicamente calcolabili"

Un risultato fondamentale

"Esiste un problema specifico che è algoritmicamente irrisolvibile"

Il problema della terminazione

Dato un programma e un input, stabilire se il programma termina su quell'input.

Teorema: il problema della terminazione è indecidibile



- Procediamo per contraddizione
- Supponiamo di avere un oracolo che risolve il problema:
  - `oracolo(programma, x)` che ritorna `True` se `programma(x)` termina, `False` altrimenti
- Usiamo l'oracolo per scrivere un nuovo programma:

```
def mr_smith(programma):
    if oracolo(programma, programma):
        while True:
            print('Sono Mr. Smith, risolvo problemi')
    else:
        print('Sono Mr. Smith, risolvo problemi')
```

- Cosa succede se eseguiamo `mr_smith(mr_smith)`?

Comprendere la dimostrazione

1. `oracolo(programma, x)` ritorna `True` esattamente quando `programma(x)` termina
   a. Banale: abbiamo assunto che l'oracolo esista e risolva il problema della terminazione
   b. `programma` è un qualsiasi programma e `x` è un qualsiasi input
2. `mr_smith` va in loop con input `programma` esattamente quando `programma(programma)` termina
   a. Cosa è successo a `x`?
   b. `x` è solo una sequenza di bit, come anche `programma`. Tutto ciò che stiamo facendo è definire quale sequenza di bit dare in input al programma.
3. `mr_smith` termina su input `mr_smith` esattamente quando `mr_smith` va in loop su input `mr_smith`
   a. Questa è la contraddizione.

Conclusioni (1)

- Esistono problemi algoritmicamente irrisolvibili'
- Problemi di interesse non solo teorico, ma anche pratico
- Esempio: Verifica del software
  - verificare che un programma è corretto non è risolvibile algoritmicamente

"O la matematica è troppo grande per la mente umana oppure la mente umana è più di una macchina."
Kurt Gödel