RAPPRESENTAZIONE MATEMATICA DI OGGETTI

Rappresentazione matematica di oggetti

Alfabeto: insieme finito di caratteri o simboli

Un oggetto è rappresentato da una sequenza ordinata di caratteri dell'alfabeto.

- A oggetti diversi corrispondono sequenze diverse.
- Il numero di oggetti che si possono rappresentare non ha limiti.

Alfabeti e sequenze

Alfabeto Γ, $|\Gamma| = s$

N oggetti da rappresentare.

- d(s, N): lunghezza della sequenza più lunga che rappresenta un oggetto dell'insieme.
- $d_{min}(s, N)$: valore minimo di d(s, N) tra tutte le rappresentazioni possibili.
- Un metodo di rappresentazione è tanto migliore, tanto più d(s, N) si avvicina a $d_{min}(s, N)$.

Rappresentazione unaria

$s = 1$, $\Gamma = \{0\}$

le sequenze di rappresentazioni possono essere solo ripetizioni dello 0

$\Rightarrow \quad d_{min}(1,N) = N$

rappresentazione estremamente sfavorevole

0, 00, 000, ..., 00000, etc.

Rappresentazione binaria

$s = 2$, $\Gamma = \{0,1\}$

- Per ogni $k \geq 1$, 2^k sequenze di lunghezza k.
- Il numero totale di sequenze lunghe da 1 a k è dato da

$$\sum_{i=1}^{k} 2^i = 2^{k+1} - 2$$

- N oggetti da rappresentare

$\Rightarrow \quad 2^{k+1} - 2 \geq N$

$\Rightarrow \quad k \geq \log_2 (N+2) - 1$

Rappresentazione binaria

$d_{min}(2,N)$: minimo intero k che soddisfa tale relazione

$\Rightarrow \quad d_{min}(2,N) = \lceil \log_2(N+2) - 1 \rceil$,

$\Rightarrow \quad \lceil \log_2 N \rceil - 1 \leq d_{min}(2,N) \leq \lceil \log_2 N \rceil$

Rappresentazione binaria

$\lceil \log_2 N \rceil$ caratteri binari sono sufficienti per costruire **N** sequenze differenti:

$N = 7, \quad \lceil \log_2 7 \rceil = 3$

0, 1, 00, 01, 10, 11, 000

Si possono costruire **N** sequenze differenti tutte di $\lceil \log_2 N \rceil$ caratteri.

$N = 7$

000, 001, 010, 011, 100, 101, 110

Rappresentazioni s-arie

- Si possono costruire N sequenze differenti con $\lceil \log_s N \rceil$ caratteri.
- Si possono costruire N sequenze differenti tutte di $\lceil \log_s N \rceil$ caratteri.

Esempio

$\Gamma = \{0,1,2,\ldots,9\}$

con $\lceil \log_{10} 1000 \rceil = 3$ caratteri:

1000 sequenze (dalla 000 alla 999)

Rappresentazioni

Usare sequenze della stessa lunghezza è vantaggioso:

- per concatenare sequenze di lunghezza diversa è necessario inserire una marca di separazione tra l'una o l'altra.
- questi accorgimenti non sono necessari se la lunghezza delle sequenze è unica e nota.

Rappresentazioni efficienti

Rappresentazioni che usano un numero massimo di caratteri di ordine logaritmico nella cardinalità N dell'insieme da rappresentare.

- l'alfabeto deve contenere almeno 2 caratteri

Rappresentazione di interi

- La notazione posizionale per rappresentare i numeri interi è una rappresentazione efficiente, indipendentemente dalla base $s \geq 2$ scelta.
- Un intero N è rappresentato con un numero d di cifre t. c.

$$\lceil \log_s N \rceil \leq d \leq \lceil \log_s N \rceil + 1$$

- Riduzione logaritmica tra il valore N di un numero e la lunghezza d della sua rappresentazione

TEORIA DELLA CALCOLABILITÀ

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 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

Insiemi numerabili

- Due insiemi A e B hanno lo stesso numero di elementi

$\Leftrightarrow$

si può stabilire una corrispondenza biunivoca tra i loro elementi

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

$\Leftrightarrow$

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

Insiemi numerabili

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

$$a_1, a_2, \ldots, a_n, \ldots$$

Insiemi numerabili: esempi

Insieme dei numeri naturali N

Insieme dei numeri interi Z

$n \leftrightarrow 2\,n + 1 \qquad n \geq 0$

$n \leftrightarrow 2\,|n| \qquad n < 0$

0, -1, 1, -2, 2, -3, 3, -4, 4, …

Insieme dei numeri naturali pari

$2n \leftrightarrow n$ 0, 2, 4, 6, 8, …

Insiemi numerabili: esempi

Insieme delle stringhe finite di simboli di un alfabeto finito

Enumerazione delle sequenze

Si vogliono elencare in un ordine ragionevole le sequenze di lunghezza finita costruite su un alfabeto finito

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.

Ordinamento canonico

- si stabilisce un ordine tra i caratteri dell'alfabeto
- 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.

Esempio

Γ = {a,b,c, ..., z}

a, b, c, ..., z,

aa, ab, ..., az, ba, ..., bz, ..., za, ..., zz,

aaa, aab, ... , baa, ..., zaa, ... , zzz,

aaaa, ...

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 che occupa la posizione n nell'elenco**.

Osservazioni

La numerazione delle sequenze è possibile perché esse sono di **lunghezza finita** anche se illimitata

cioè per un qualunque intero d scelto a priori esistono sequenze di lunghezza maggiore di d

Per sequenze di lunghezza infinita la numerazione non è possibile.

Insiemi non numerabili

Sono tutti gli insiemi non equivalenti a N

Esempi:

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

Problemi computazionali

L'insieme dei problemi computazionali **NON** è numerabile

Problemi e funzioni

Un problema computazionale può essere visto come una funzione matematica che associa ad ogni insieme di dati, espressi da k numeri interi, il corrispondente risultato, espresso da j numeri interi

$$f: N^k \rightarrow N^j$$

L'insieme delle funzioni $f: N^k \rightarrow N^j$ NON è numerabile

Diagonalizzazione

F = { funzioni f | $f: N \rightarrow \{0,1\}$ }

ogni $f \in F$ può essere rappresentata da una sequenza infinita

x	0	1	2	3	4	...	n	...
$f(x)$	0	1	0	1	0	...	0	...

o, se possibile, da una regola finita di costruzione

$$f(x) = \begin{cases} 0 & x \ pari \\ 1 & x \ dispari \end{cases}$$

Diagonalizzazione

Teorema

L'insieme F non è numerabile

Dim.

- Per assurdo, F sia numerabile
- Possiamo enumerare ogni funzione

  assegnare ad ogni $f \in F$ un numero progressivo nella numerazione, e costruire una tabella (infinita) di tutte le funzioni

Diagonalizzazione

x	0	1	2	3	4	5	6	7	8	...
$f_0(x)$	1	0	1	0	1	0	0	0	1	...
$f_1(x)$	0	0	1	1	0	0	1	1	0	...
$f_2(x)$	1	1	0	1	0	1	0	0	1	...
$f_3(x)$	0	1	1	0	1	0	1	1	1	...
$f_4(x)$	1	1	0	0	1	0	0	0	1	...
...				...						...

Diagonalizzazione

Consideriamo la funzione $g \in F$

$$g(x) = \begin{cases} 0 & f_x(x) = 1 \\ 1 & f_x(x) = 0 \end{cases}$$

g non corrisponde ad alcuna delle f_i della tabella:

differisce da tutte nei valori posti sulla diagonale principale

Diagonalizzazione

x	0	1	2	3	4	5	6	7	8	...
$f_0(x)$	1	0	1	0	1	0	0	0	1	...
$f_1(x)$	0	0	1	1	0	0	1	1	0	...
$f_2(x)$	1	1	0	1	0	1	0	0	1	...
$f_3(x)$	0	1	1	0	1	0	1	1	1	...
$f_4(x)$	1	1	0	0	1	0	0	0	1	...
...										

Diagonalizzazione

x	0	1	2	3	4	5	6	7	8	...
$f_0(x)$	1	0	1	0	1	0	0	0	1	...
$f_1(x)$	0	0	1	1	0	0	1	1	0	...
$f_2(x)$	1	1	0	1	0	1	0	0	1	...
$f_3(x)$	0	1	1	0	1	0	1	1	1	...
$f_4(x)$	1	1	0	0	1	0	0	0	1	...
...										
g(x)										

Diagonalizzazione

x	0	1	2	3	4	5	6	7	8	...
$f_0(x)$	**1**	0	1	0	1	0	0	0	1	...
$f_1(x)$	0	0	1	1	0	0	1	1	0	...
$f_2(x)$	1	1	0	1	0	1	0	0	1	...
$f_3(x)$	0	1	1	0	1	0	1	1	1	...
$f_4(x)$	1	1	0	0	1	0	0	0	1	...
...										
g(x)	**0**									

Diagonalizzazione

x	0	1	2	3	4	5	6	7	8	...
$f_0(x)$	**1**	0	1	0	1	0	0	0	1	...
$f_1(x)$	0	**0**	1	1	0	0	1	1	0	...
$f_2(x)$	1	1	0	1	0	1	0	0	1	...
$f_3(x)$	0	1	1	0	1	0	1	1	1	...
$f_4(x)$	1	1	0	0	1	0	0	0	1	...
...										
g(x)	**0**	**1**								

Diagonalizzazione

x	0	1	2	3	4	5	6	7	8	...
$f_0(x)$	**1**	0	1	0	1	0	0	0	1	...
$f_1(x)$	0	**0**	1	1	0	0	1	1	0	...
$f_2(x)$	1	1	**0**	1	0	1	0	0	1	...
$f_3(x)$	0	1	1	0	1	0	1	1	1	...
$f_4(x)$	1	1	0	0	1	0	0	0	1	...
...										
g(x)	**0**	**1**	**1**							

Diagonalizzazione

x	0	1	2	3	4	5	6	7	8	...
$f_0(x)$	**1**	0	1	0	1	0	0	0	1	...
$f_1(x)$	0	**0**	1	1	0	0	1	1	0	...
$f_2(x)$	1	1	**0**	1	0	1	0	0	1	...
$f_3(x)$	0	1	1	**0**	1	0	1	1	1	...
$f_4(x)$	1	1	0	0	1	0	0	0	1	...
...										
g(x)	**0**	**1**	**1**	**1**						

Diagonalizzazione

x	0	1	2	3	4	5	6	7	8	...
$f_0(x)$	**1**	0	1	0	1	0	0	0	1	...
$f_1(x)$	0	**0**	1	1	0	0	1	1	0	...
$f_2(x)$	1	1	**0**	1	0	1	0	0	1	...
$f_3(x)$	0	1	1	**0**	1	0	1	1	1	...
$f_4(x)$	1	1	0	0	**1**	0	0	0	1	...
...										
g(x)	**0**	**1**	**1**	**1**	**0**	.	.	.		

Diagonalizzazione

Per assurdo: $\exists\ j$ t.c. $g(x) = f_j(x)$

allora $g(j) = f_j(j)$, ma per definizione

$$g(j) = \begin{cases} 0 & f_j(j) = 1 \\ 1 & f_j(j) = 0 \end{cases}$$

cioè $g(j) \neq f_j(j)$

$\Rightarrow$ contraddizione!!!

Diagonalizzazione

Per qualunque numerazione scelta, esiste sempre almeno una funzione esclusa

$\Rightarrow$ F non è numerabile

Si possono considerare linee arbitrarie che attraversano la tabella toccando tutte le righe e tutte le colonne esattamente una volta, e definire funzioni che assumono in ogni punto un valore opposto a quello incontrato sulla linea

Esistono infinite funzioni di F escluse da qualsiasi numerazione

Conclusione

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

  $f: N \rightarrow N$

  $f: N \rightarrow R$

  $f: R \rightarrow R$

  $f: N^k \rightarrow N^j$

$\Rightarrow$

L'insieme dei problemi computazionali non è numerabile

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

$$|\{Problemi\}| >> |\{Algoritmi\}|$$

$$\Rightarrow$$

Esistono problemi privi di un corrispondente algoritmo di calcolo

UN PROBLEMA INDECIDIBILE: il problema dell'arresto

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

È un problema posto in forma decisionale

Arresto: {Istanze} → {0,1}

Per i problemi decisionali, la calcolabilità è in genere chiamata **decidibilità**

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

```
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;
```


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

Algoritmo GOLDBACH

- Trova il più piccolo numero intero pari (maggiore o uguale a 4) che **NON** sia la somma di due numeri primi
- Si arresta quando trova $n \geq 4$ che NON è la somma di due primi

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

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)

ARRESTO(A,A) = 1

$\Leftrightarrow$

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 = ARRESTO(A,A) = 0

A(A) non termina

DIMOSTRAZIONE

Cosa succede calcolando PARADOSSO(PARADOSSO)?

PARADOSSO(PARADOSSO) **termina**

x = ARRESTO(PARADOSSO, 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

Il problema dell'arresto è indecidibile

Osservazione

Non può esistere un algoritmo che decida in tempo finito se un algoritmo arbitrario termina la sua computazione su dati arbitrari

ciò non significa che non si possa decidere in tempo finito la terminazione di algoritmi particolari

il problema è indecidibile su una coppia <A,D> scelta arbitrariamente

Osservazione

L'algoritmo ARRESTO costituirebbe uno strumento estremamente potente

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

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

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?

Modelli di calcolo

I linguaggi di programmazione esistenti sono tutti equivalenti?

Ce ne sono alcuni più potenti e/o più semplici di altri?

Ci sono algoritmi descrivibili in un linguaggio, ma non in un altro?

È possibile che problemi oggi irrisolvibili possano essere risolti in futuro con altri linguaggi o con altri calcolatori?

La teorie della calcolabilità e della complessità dipendono dal modello di calcolo?

La tesi di Church-Turing

Tutti i (ragionevoli) modelli di calcolo

- *risolvono esattamente la stessa classe di problemi*
- dunque si equivalgono nella possibilità di risolvere problemi, pur operando con diversa efficienza

La **TESI** di Church-Turing

La decidibilità è una proprietà del problema

Incrementi qualitativi alla struttura di una macchina, o alle istruzioni di un linguaggio di programmazione, servono ***solo*** a

- abbassare il tempo di esecuzione
- rendere più agevole la programmazione