

# Indecidibilità e Casualità

- Alan Turing '37 ←
- Andrey Kolmogorov '60s (Chaitin)

Distinzione tra algoritmo e dati? No  
Sequenza di bit → dato input  
                                    → programma/algoritmo

```
using namespace std;  
int main () {  
    cout << "ciao" << endl;  
}
```

$\langle A, D \rangle$

algoritmo, dato  
A D  
come  
input

$\langle A, A \rangle$   
↻

```

1 CongetturaGoldbach( ):
2   n = 2;
3   DO {
4     n = n + 2;
5     controesempio = TRUE;
6     FOR (p = 2; p <= n-2; p = p + 1) {
7       q = n - p;
8       IF (Primo(p) && Primo(q)) controesempio = FALSE
9     }
10  } WHILE (!controesempio);
11  RETURN n;

```

$\forall n \geq 4, n \text{ pari}$   
 $\exists p, q \text{ primi (non necessariamente)} \\ \text{distinti}$   
 t.c.  $n = p + q$

CONGETTURA VERA  $\nleftrightarrow$  programma NON termina

Riduzione: riformulato la congettura suddetta  
 come un problema di terminazione del programma

$i = 0; j = n;$   
 while (  $i < j$  )  $i++$

↳ facile vedere la terminazione

è un  
 problema  
 aperto

$TERMINA(A, D) \begin{matrix} \text{SI} \\ \text{NO} \end{matrix}$  esiste un tale algoritmo?

SI se algoritmo A con input D termina

NO altrimenti

Eseguire A direttamente su D non funziona  
nel caso che A non termini

Turing '37 (ispirato da Gödel)

$TERMINA(, )$  NON esiste

TERMINA(A, A) è perfettamente legittimo  
algoritmo dato

Ipotesi d'assunto: TERMINA( ) esiste

Nuovo algoritmo che posso costruire usando TERMINA( )

PARADOSSO(X):

while (TERMINA(X, X))

; // istruzione vuota

PARADOSSO(PARADOSSO) termina?

- SÌ  $\Rightarrow$  Conditione while è falsa,  $X = P \Rightarrow \text{TERMINA}(PP) = \text{NO}$
- NO  $\Rightarrow$  " " " " VERA  $\Rightarrow P(P) \xRightarrow{\text{TERMINA}} P(P)$  non termina



$P(P)$  termina  $\Leftrightarrow P(P)$  non termina

ASSURDO: l'unico punto debole è l'assunzione che  $TERMINA(A,D)$  esiste.

Non tutti i problemi computazionali hanno un algoritmo come soluzione

$\Pi: \mathbb{N} \rightarrow \mathbb{N}$

BIEZIONE

lex-order

Quanti problemi computazionali  $\Pi$  abbiamo?

# funzioni:  $\mathbb{N} \rightarrow \mathbb{N}$

↳ cardinalità reali

1	0	0	1N
2	00	1	
3	01	2	
4	10	3	
5	11	4	
6	000	5	
	001	6	
	010		
	011		
	100		
	111		

Quanti algoritmi / programmi?

Se prendiamo la loro codifica in binario,  
non superano il numero di sequenze binarie  $\{0,1\}^*$

# algoritmi  
# programmi = cardinalità dei naturali

Quindi ci sono più problemi computazionali (i reali)  
che algoritmi / programmi (i naturali)

Problema è  
indecidibile

Contributo di Turing: eccone uno!

- Fermata (Haltin)

$\forall D: A_1(D) = A_2(D)$

- Equivalenza di 2 programmi :  $A_1, A_2$  dire se

Cos'è costante?

010101010101  
n volte 01

$2n$  bit

NO c'è un pattern: una qualche regola sottostante

A: 

```
for (i=0; i<n; i++)  
    cout << "01";  
cout << endl;
```

$\langle A, n \rangle$

algorithm data

$C + \log_2 n$  bit

costante  $C$

010101...01 = X  
n volte

$2n$  bit

$$c + \lg_2 n < 2n \quad \text{per } c > 0 \quad \text{e } n > n_0$$

KOLMOGOROV COMPLEXITY, sequenza  $x \in \{0,1\}^*$

$K_L(x)$  = più piccolo algoritmo  $A$  con input  $y$  t.c.  $A(y) = x$

più piccola codifica binaria di  $\langle A, y \rangle$  t.c.  $A(y) = x$   
(nel linguaggio di programmazione  $L$ )

non possiamo usare in  $L$  meno di  $K_L(x)$  bit per generare  $x$

(definizione robusta perché  $K_L(x)$  e  $K_{L'}(x)$  differiscono solo di una costante additiva)

$K(x)$  senza specificare  $L$

ZIP: nessun può comprimere in meno di  $K(x)$  bit,  
a meno di una costante addizionale

CONTRIBUTO DI KOLMOGOROV: collegare CASUALITA' con  
COMPRESSIBILITA'

$x$  è casuale (RANDOM) se  $K(x) \geq |x| - c$ , costante  $c$

non esiste regola/pattern/algoritmo/programma  
che riesca a generare  $x$  in modo più succinto,  
se non quello di scrivere  $x$  "esplicitamente"

BUONA e CATTIVA NOTIZIA:

La stuprante maggioranza di stringhe binarie è RANDOM  
ma è indecidibile stabilire se  $x$  è RANDOM.

Fatto 1  $\forall n \exists x \in \{0,1\}^n$  t.c.  $k(x) \geq n$

$|\{0,1\}^n| = 2^n$  per induzione

$$S = \{x \in \{0,1\}^n : k(x) < n\}$$

$|S| \leq \begin{matrix} \# \text{ sequenze} \\ \text{binarie} \text{ di} \\ \text{lunghe} \\ \text{al più } n-1 \end{matrix} \quad \begin{matrix} \uparrow \\ \langle A, \gamma \rangle \text{ t.c. richiesto } \leq n-1 \text{ bit} \end{matrix}$

$$|S| \leq 2^0 + 2^1 + 2^2 + \dots + 2^{n-1} = 2^n - 1$$

$\Rightarrow \exists$  almeno una  $x$  t.c.  $k(x) \geq n$

Fatto 2  $\forall n \forall c$  : prendete una stringa  $x \in \{0,1\}^n$  scelta a caso

$$\Pr(k(x) \geq \underline{n-c}) > 1 - \frac{1}{2^c}$$

$$S = \{x \in \{0,1\}^n : k(x) < n-c\}$$

$$|S| \leq 2^0 + 2^1 + \dots + 2^{n-c-1} = 2^{n-c} - 1$$

$$\Pr(k(x) < n-c) = \frac{|S|}{2^n} \leq \frac{2^{n-c} - 1}{2^n} = \frac{1}{2^c} - \frac{1}{2^n} < \frac{1}{2^c}$$

$$\Pr(k(x) \geq n-c) = 1 - \Pr(k(x) < n-c) > 1 - \frac{1}{2^c} \quad \square$$

$$\langle A, n \rangle = 01011101101$$

$$0 \leftrightarrow 00$$

$$1 \rightarrow 11$$

$$\begin{array}{c|c} \underbrace{00} & \underbrace{11} \\ \underbrace{00} & \underbrace{11} \\ \underbrace{11} & \underbrace{11} \\ \underbrace{11} & \underbrace{11} \\ \underbrace{11} & \underbrace{11} \end{array} \quad \begin{array}{c|c} \underbrace{00} & \underbrace{11} \\ \underbrace{11} & \underbrace{00} \\ \underbrace{11} & \underbrace{00} \\ \underbrace{11} & \underbrace{00} \\ \underbrace{11} & \underbrace{00} \end{array}$$

A

n

$$01 \text{ --- } 10 \text{ --- } 01$$

$$\begin{array}{c} 01 \quad 01 \\ < \quad > \end{array}$$

$$\begin{array}{c} 10 \\ / \end{array}$$

② online lex

$$\begin{array}{ccccccccc} 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & 010, & \dots, & 111, & 0000, & 0001, & \dots, & 1111, & 00000, & \dots \\ \underbrace{1} & \underbrace{2} & \underbrace{3} & \underbrace{4} & \underbrace{5} \end{array}$$

esiste un algoritmo L che genera tutte le infinite stringhe binarie in ordine lessicografico



Sia  $R = \{x \in \{0,1\}^* : K(x) \geq |x|\}$  l'insieme delle stringhe random secondo Kolmogorov

► L'appartenenza a  $R$  è un problema indecidibile (come il problema della fermata di Turing)

Per assurdo, supponiamo esista un algoritmo  $A$  per decidere  $R$ .

Costruiamo un algoritmo  $B$ :

① usa l'algoritmo  $L$  per generare una stringa  $x$  alla volta in ordine lessicografico

② data ogni  $x$  presa dal passo ①, eseguiamo  $A(x)$

Per stabilire se  $x \in R$  in tempo finito (per h.p. di assurdo).

$A(x)$   $\forall \forall \forall \forall \forall \forall \forall \forall \forall \forall$   
 $x = 0, 1, 00, 01, 10, 11, 000, 001, 010, \dots, 111, 0000, 0001, \dots, 1111, 00000, \dots$

1 2 3 4 5

Il passo ② si ferma la prima volta che  $A(x)$  risponde che  $x \in R$   
(nota -  $R \neq \emptyset$  per il Fatto 1)

$\Rightarrow B$  termina perché  $A$  termina e  $R \neq \emptyset$

$B$  trova  $\min_{\leq_L} R$  dove  $\leq_L$  è l'ordine lessicografico

$\langle B, n \rangle$  come sopra, solo che considera le stringhe  $x$  t.c.  $|x|=n$   
(Fatto 1: esiste sempre almeno una)  $x \in R$

Sia  $s_n$  la sequenza binaria che codifica  $\langle B, n \rangle$

$$K(s_n)? \quad |s_n| \geq K(s) \geq n$$

$s_n \leq \lg n + c$  bit  $\Rightarrow$  contraddizione perché  $\lg n + c \geq n$  vale solo per pochi valori