LA TEORIA DELLA COMPLESSITA' COMPUTAZIONALE
I limiti teorici dell'uso dei computer

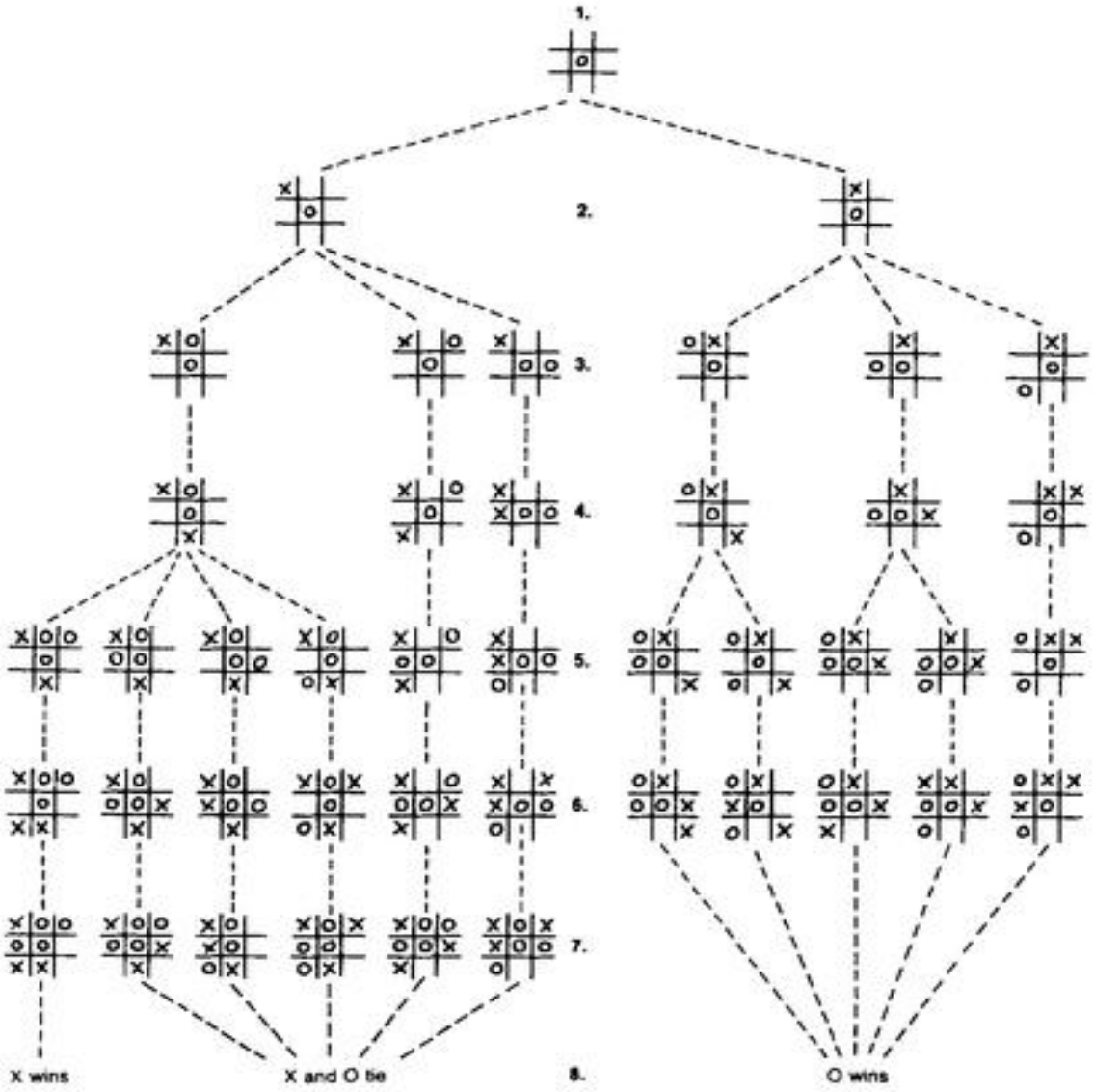
Carlo Toffalori (Camerino)

Summer School "La matematica incontra le altre scienze"
San Pellegrino Terme, 8 settembre 2014

Un teorema di un secolo fa a proposito degli scacchi: Ernst Zermelo, 1912) e l'avvio della teoria dei giochi (economici)

Come nel Tris…

L’albero del Tris



1. The first move by O.
2. Possible first moves by X.
3. Possible second moves by O.
4. Possible second moves by X.
5. Possible third moves by O.
6. Possible third moves by X.
7. Possible fourth moves by O.
8. Outcomes.

- Un errore fondamentale da evitare per X
- Un errore fondamentale da evitare per O

Dopo di che la partita perfetta, immune da errori, finisce in pari

Il teorema di Zermelo: ***c'è (almeno) una partita perfetta anche per gli scacchi!*** Ma allora perché si continua a giocare a scacchi?

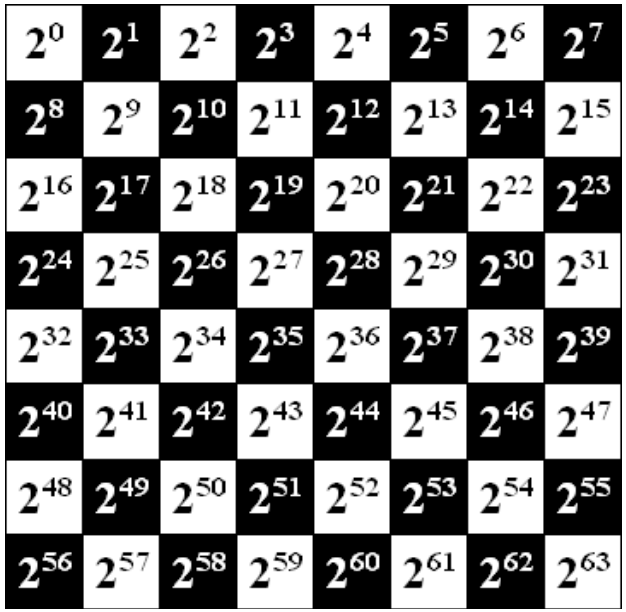
Kasparov contro Deep Blue, l'uomo contro la macchina

Negli scacchi mosse di apertura, 10^{43} posizioni, 10^{120} partite possibili…

Da notare: oggi un calcolatore esegue al più 10^{17} operazioni all'anno

Dunque: quanti anni per completare tutte le partite e individuare quelle perfette?

Pratica e grammatica, matematica e teologia…



Sissa e la leggenda della nascita degli scacchi

$$S = 1 + 2^1 + 2^2 + 2^3 + \ldots + 2^{62} + 2^{63}$$
$$2\,S = \quad 2^1 + 2^2 + 2^3 + \ldots + 2^{62} + 2^{63} + 2^{64}$$

e sottraendo membro a membro dal basso vero l'alto

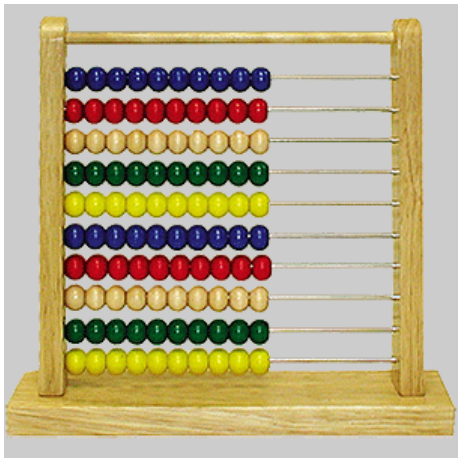
$$S = 2\,S - S = \mathbf{2^{64} - 1}$$

Limiti all'opera dei computer? Ci sono problemi

- ***difficili*** da risolvere?
- o ***impossibili*** da risolvere?

Ma che cosa significa

- ***difficile*** da risolvere?
- o ***impossibile*** da risolvere?



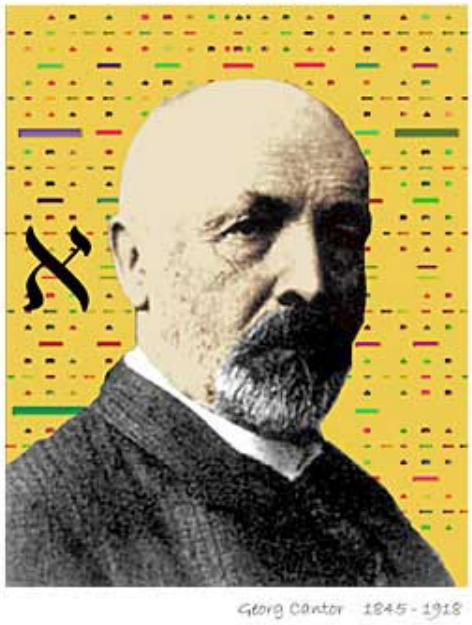
Una premessa doverosa, per chiarire il contesto: ci interessano quei problemi che si possono esprimere tramite

- i numeri naturali 0, 1, 2, 3, 4, …
- oppure i numeri interi …, – 4, – 3, – 2, – 1, 0, +1, +2, +3, +4, …
- oppure i numeri razionali $\frac{M}{N}$ con M, N interi **primi tra loro**,

dunque: sommare, sottrarre, moltiplicare, dividere, riconoscere i primi dai composti, ***ma non solo***…

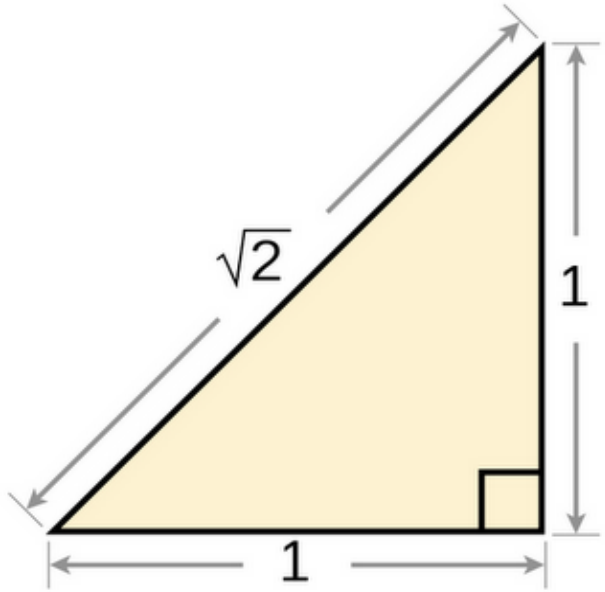
Dicevano i pitagorici: ***"tutto è numero"*** (naturale, intero, razionale)

In effetti, procedimenti di numerazione famosi: Georg Cantor, Kurt Gödel, …



Eppure… *il teorema di Pitagora*

- La sua equazione $\boldsymbol{x^2 + y^2 = z^2}$
- Il caso più semplice: il triangolo rettangolo isoscele, $x = y = 1$, $\boldsymbol{z^2 = 2}$



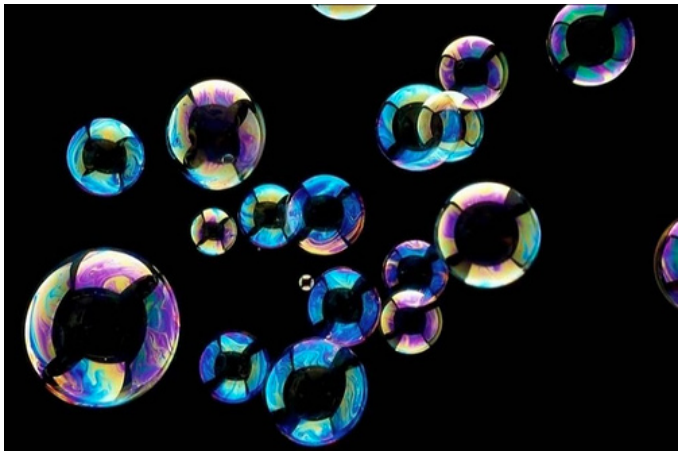
Per $x = y = 1$, z è la radice di 2, un numero reale irrazionale ***infinitamente lungo***, da guadagnarsi cifra per cifra 1,414213562373095048801688724209 69…

Perché $\sqrt{2}$ non può essere un razionale $\frac{M}{N}$ con *M*, *N* interi positivi **primi tra loro**

- Altrimenti $\frac{M^2}{N^2} = 2$
- Quindi $M^2 = 2\,N^2$
- Quindi, per l'unicità della decomposizione in fattori primi degli interi, *M* è pari e $M^2 = M \times M$ è divisibile per 4
- Quindi, sempre per l'unicità della decomposizione in fattori primi degli interi, anche *N* è pari

Ma *M* e *N* sono primi tra loro…

Dunque i Pitagorici si sbagliavano…



Lo sviluppo decimale della radice di 2

- troppo lungo per essere dominato da un computer,
- se non altro dotato di semplici algoritmi per essere calcolato cifra per cifra

E comunque: oggi

l'informazione è numero

perché nei computer tutto – input, output, programmi – si rappresenta con stringhe di numeri naturali (addirittura di 0 e 1).

La prima domanda: ci sono problemi impossibili da risolvere?

Un dibattito scientifico negli anni '30 (Hilbert, Church, Gödel, Turing, altri…)

- Che cosa si può *calcolare*?
- Che significa *calcolare*?
- Chi è delegato a *calcolare*? Chi, o che cosa, è il *calcolatore*?

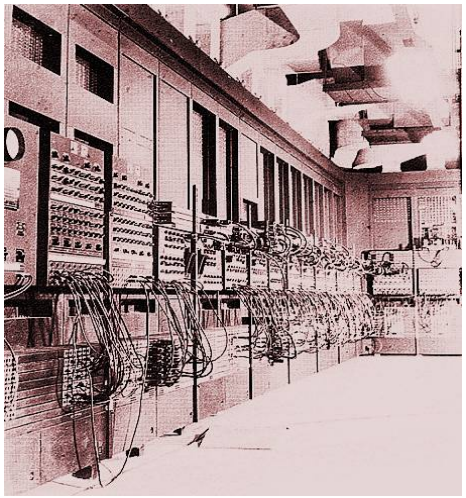
La risposta del 1936 e la nascita dell'informatica moderna

- Il lambda-calcolo di Church
- Le funzioni parziali ricorsive di Gödel, Kleene e altri
- Ulteriori proposte equivalenti alle precedenti
- Soprattutto ***la macchina di Turing***

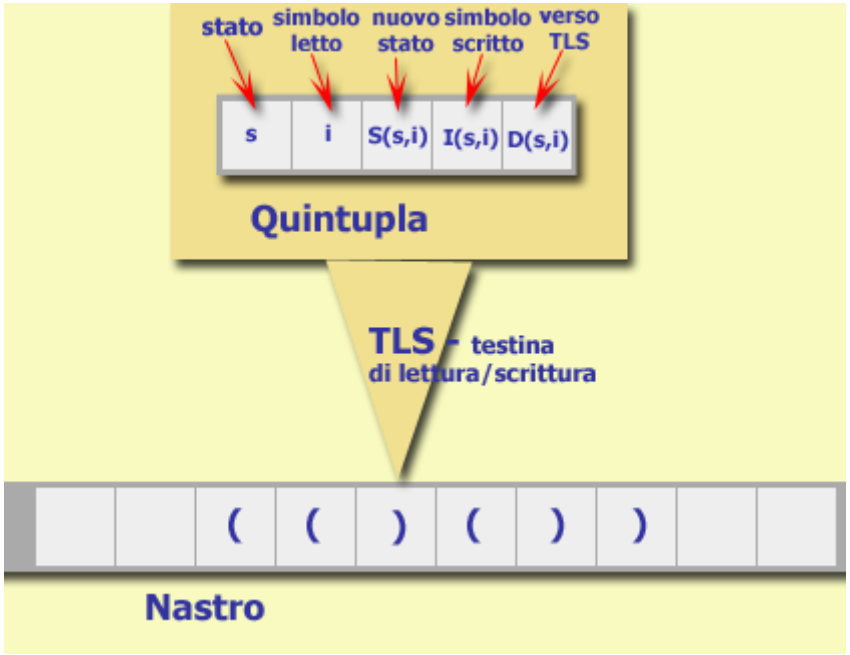
Alan Turing, 1936-37, *On **computable numbers** with an application to the **Entscheidungsproblem***, Proceedings London Mathematical Society

Propone il primo modello teorico di calcolatore moderno: appunto, la *Macchina di Turing.*

Da notare: il primo computer elettronico moderno, l'ENIAC di John von Neumann, risale al 1946.



La *Macchina di Turing*: la madre di tutti i calcolatori moderni.



Soprattutto…

La Tesi di Church e Turing *(1936): è calcolabile* ***esattamente quello*** *che una Macchina di Turing sa calcolare.*

Per molti versi valida ancora oggi… le motivazioni a suo sostegno

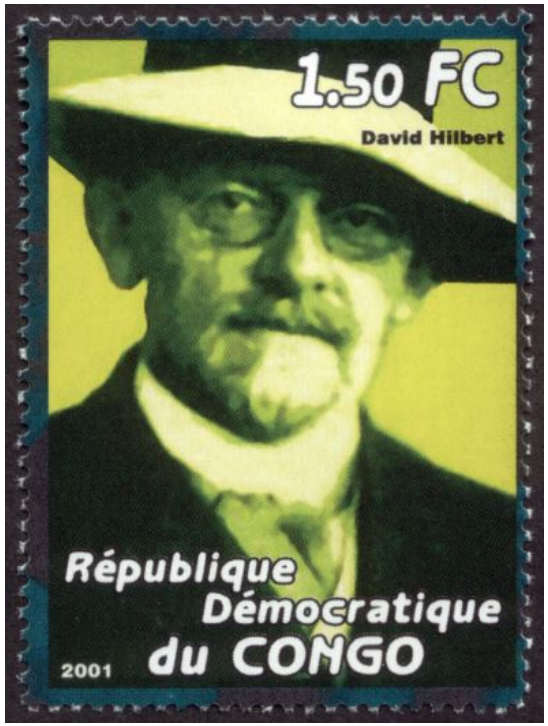
- l'assenza di smentite sperimentali,
- l'equivalenza con gli altri approcci alla calcolabilità sopra elencati,
- la simulazione meccanica della mente (*l'impiegato diligente*).

Tornando alla tesi di Church e Turing…

Sulla sua base: ***problemi (matematici e informatici) che NON si possono risolvere.***

1. L'***Entscheidungsproblem*** (parola tedesca di spavento, in italiano il *problema della decisione*)
2. I ***numeri calcolabili*** (oggi ***reali ricorsivi***, sono i numeri reali che, come radice di 2, hanno un algoritmo capace di calcolare il loro sviluppo decimale; ma *esistono miriadi di numeri reali privi di un tale algoritmo…*)
3. Il ***problema dell'arresto*** delle macchine di Turing
4. Un Entscheidungsproblem ristretto all'***aritmetica dei naturali con le operazioni elementari*** **+,** × (e volendo – e :).

Un altro esempio famoso: il ***Decimo Problema di Hilbert***



David Hilbert, Parigi, 1900: *determinare un procedimento per stabilire,*

- *per ogni equazione a coefficienti interi (di qualunque grado, in qualunque numero di variabili),*
- *se questa equazione ha o no soluzioni intere.*

Per intendersi

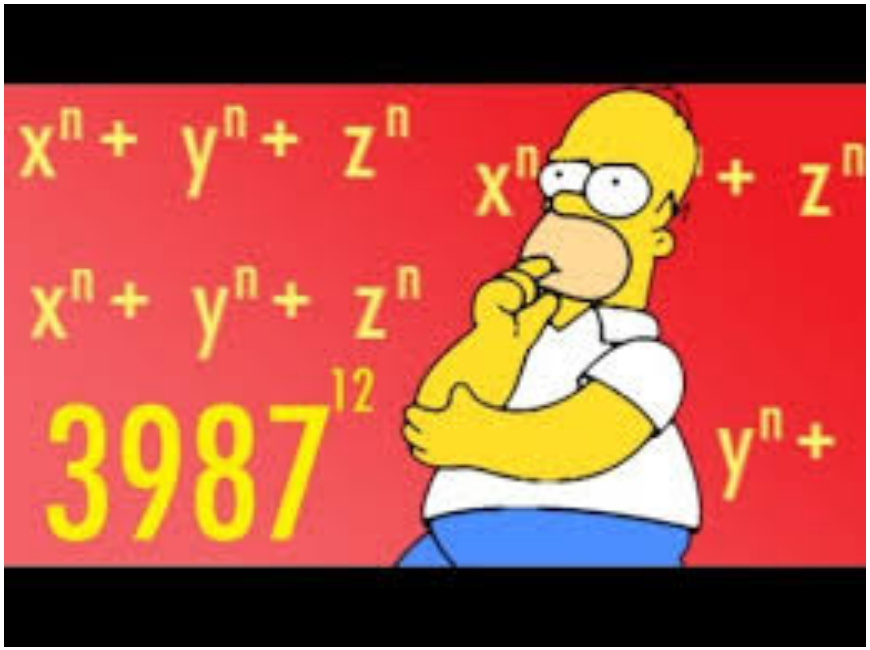
$$2x + 4 = 0, \quad x^2 - 1 = 0, \quad x^2 + y^2 - 1 = 0, \ldots$$
$$2x + 5 = 0, \quad x^2 - 2 = 0, \quad x^2 + y^2 + 1 = 0, \ldots$$

sono tutte equazioni a coefficienti interi, ma

- quelle della prima riga hanno anche soluzioni intere, rispettivamente -2, ± 1, $(\pm 1, 0)$ e $(0, \pm 1)$,
- quelle della seconda riga no,

eppure la differenza tra le prime e le seconde è quasi impalpabile.

C'è un algoritmo generale che sa distinguere le prime equazioni dalle seconde?



Casi discussi e famosi (a testimoniare la difficoltà del problema)

- l'equazione $x^2 - 2 = 0$ che determina $\sqrt{2}$,
- l'equazione $x^2 + 1 = 0$ che genera i e i numeri complessi,
- l'equazione di Bombelli $x^3 - 15\,x - 4 = 0$,
- l'equazione del teorema di Pitagora $x^2 + y^2 = z^2$
- le equazioni dell'Ultimo Teorema di Fermat $x^n + y^n = z^n$ per $n \geq 3$.

La soluzione del problema, 1970, Yuri Matijasevic, a coronamento del lavoro di Martin Davis, Julia Robinson…

… e Hilary Putnam): ***l'algoritmo richiesto non esiste!***

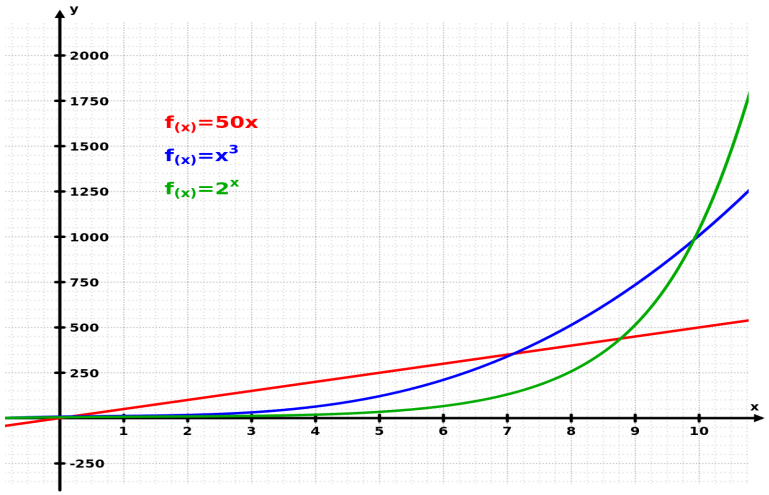
Una soluzione sorprendente, basata sul lavoro di Turing del 1936 (e sulle sue dirette conseguenze).

La seconda domanda: tra i problemi che si possono risolvere ce ne sono alcuni difficili da risolvere

Il ***costo di una soluzione*** - sempre sostenibile?

Gli scacchi ammoniscono…
… e un teorema devastante di Fischer e Rabin del 1974 conferma!

Premessa: tempi ***esponenziali*** di attesa sono improponibili (2^{64} secondi superano di gran lunga l'età del mondo)



L'***Entscheidungsproblem*** ristretto ai numeri naturali con l'addizione +

- c'è un algoritmo di soluzione (Presburger, 1929)
- ***qualunque*** algoritmo di soluzione impiega tempi che nella migliore delle ipotesi sono 2 volte esponenziali rispetto alla lunghezza dell'input

Notare: $2^{64} = 2^{2^6}$…

Samuel Beckett, *Aspettando Godot*, Vladimiro ed Estragone

Chi è il signor Godot, che *"non verrà questa sera ma di sicuro domani"*?

- Il deus ex machina Godeau de *L'affarista Mercadet* di Balzac?
- La felicità? Dio?
- Il ***computer***?

Una nuova prospettiva e i limiti della macchina: la ***Complessità Computazionale***

Galileo: *"discorrere è come correre"*

Come misurare la complessità di un problema? Dagli algoritmi "migliori" che lo risolvono

Come misurare la complessità di un algoritmo? Da chiarire preliminarmente

a) Chi esegue l'algoritmo
b) Con che parametro si misura la complessità
c) La linea di divisione *realizzabile/irrealizzabile* (nella pratica)

Le risposte (non univoche!)

a) La *macchina di Turing* (come un'Isotta Fraschini e una Ferrari)
b) Il *tempo*
c) Lo slogan *"rapido = polinomiale"*

La classe P, o PTIME *(P per "polinomiale")*
Consiste dei problemi dotati di una macchina di Turing che li risolve impiegando un tempo (= un numero di passi) ***al più polinomiale*** rispetto alla lunghezza dell'input.

La tesi di Cook e Karp (e von Neumann, Rabin, Cobham, Edmonds, …)
Un problema ammette un algoritmo rapido di soluzione ***esattamente quando*** sta in P.

Come la tesi di Church e Turing… ma molto più controversa!

- Facile convenire che *esponenziale è lento*
- Difficile convincersi che *polinomiale è rapido* (come si fa a definire rapido un algoritmo che
  - su input di lunghezza x
  - impiega tempi $x^{2^{64}}$ oppure $2^{64}x$ polinomiali rispetto a x?

Tuttavia mancano alternative credibili, suggerite da natura o teoria

Per di più una classe assai poco frequentata… Stanno in P

- i problemi di sommare e sottrarre (tempi polinomiali di grado 1),
- i problemi di moltiplicare, sottrarre o calcolare massimo comune divisore (tempi polinomiali di grado 2).

Una prestigiosa new entry. Da ricordare

- Un numero naturale N > 1 si dice ***primo*** se gli unici suoi divisori sono 1 e N, ***composto*** altrimenti

  Primi: 2, 3, 5, 7, 11, 13, 17, 19, 23, …
  Composti: 4, 6, 8, 9. 10. 12, 14, 15, 16, 18, 20, 21, 22, …

- Ogni numero naturale N > 1 si decompone nel prodotto di fattori primi, e questa rappresentazione è unica a meno dell'ordine dei fattori

$$4 = 2^2,\ 6 = 2\times 3,\ 8 = 2^3,\ 9 = 3^2,\ 10 = 2 \times 5,\ 12 = 2^3\times 3, \ldots$$

Due problemi che sorgono spontanei: per ogni numero naturale N > 1,

- riconoscere se N è primo o composto,
- decomporre N in fattori primi.

Agrawal, Kayal, Saxena, ***2002***: il problema di separare i primi dai composti è in P (tempi polinomiali di grado per ora ***almeno*** 6)

E per quanto riguarda il problema della decomposizione in fattori primi?

La parola a un esperto (H. W. Lenstra)

"Ammettiamo di avere due numeri primi $p \neq q$ e il loro prodotto $N = p \cdot q$ e di smarrire p, q in un letamaio, così che ci rimane solo N. Deve essere sentito come una sconfitta della scienza il dover ammettere che l'algoritmo più rapido per recuperare p e q è quello di cercare nel letamaio"

Ma se N è davvero $p \times q$ (o comunque composto),

- rapido suggerire p o q (o comunque un divisore),
- rapido controllare che $N = p \times q$, o comunque recuperare la decomposizione di N.

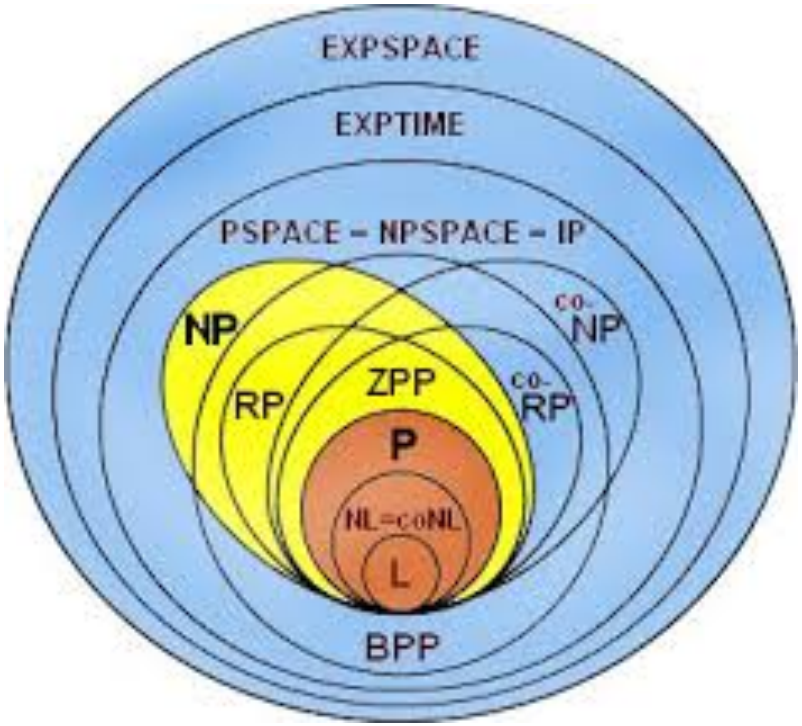
Tutto diventa rapido ***con un po' di aiuto****!*

La classe NP, o NPTIME *(N per "non deterministico")*
Consiste dei problemi per le cui istanze positive è possibile fornire in tempo al più polinomiale un suggerimento col cui aiuto si verifica in tempo al più polinomiale la risposta

Chiaramente P è contenuta in NP (i problemi rapidi da risolvere senza aiuto lo restano anche con un po' di aiuto…)

Un problema del millennio per la matematica del 2000: P = NP? Fino a che punto fortuna e aiuti sono importanti per far carriera?

L'opinione prevalente: P ≠ NP



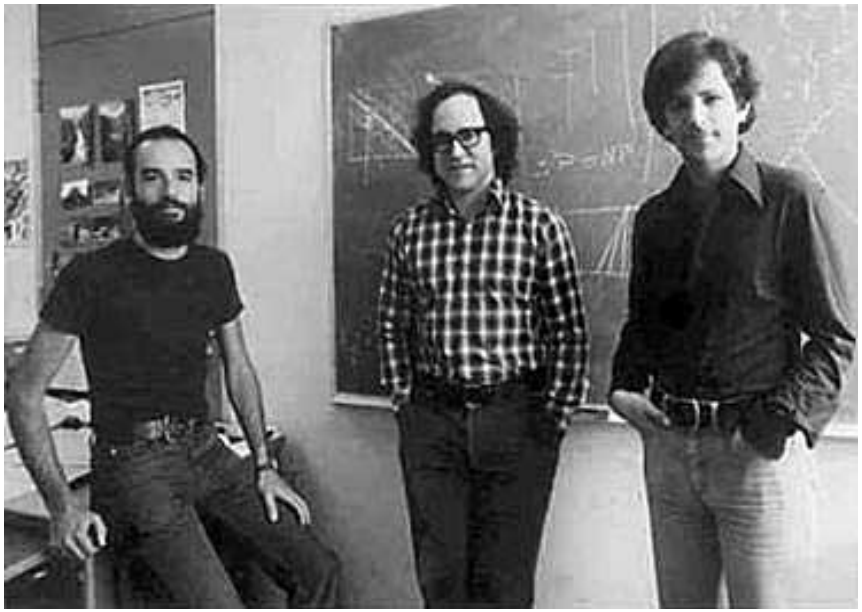
Lo zoo della complessità

a) Macchine di Turing deterministiche, non deterministiche, probabilistiche, interattive, …
b) Tempo, memoria, energia, …
c) Polinomi, logaritmi, …

Un'applicazione: la crittografia a chiave pubblica (ovvero come riciclare perfino le difficoltà).

Supponiamo di

- avere un segreto da nascondere a intrusi (un codice bancario, una comunicazione riservata, …)
- conoscere di un dato N i fattori p e q, che invece i nostri avversari ignorano
- di affidarci a p e q per conservare il nostro segreto.



La base del crittosistema RSA (Rivest, Shamir, Adleman, 1976)

Molti sviluppi da allora

2012, Shafi Goldwasser e Silvio Micali, Premio Turing (il Nobel dell'Informatica) per i contributi alla Complessità Computazionale e alla Crittografia