

# Algoritmi e computabilità

Informatica di base – a.a. 2022/2023

Silvio Peroni

[0000-0003-0530-4305](https://orcid.org/0000-0003-0530-4305)

Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia  
[silvio.peroni@unibo.it](mailto:silvio.peroni@unibo.it) – [@essepuntato](https://www.essepuntato.it) – <https://www.unibo.it/sitoweb/silvio.peroni/>



Quest'opera è distribuita con [Licenza Creative Commons Attribuzione 4.0 Internazionale](https://creativecommons.org/licenses/by/4.0/)



DIPARTIMENTO DI FILOLOGIA CLASSICA E ITALIANISTICA

# Situazioni di tutti i giorni 1

## Per la base (per una tortiera del diametro di 22 cm)

- Biscotti Digestive 240 g
- Burro 110 g

### PER LA CREMA

- Formaggio fresco spalmabile 500 g
- Panna fresca liquida 100 g
- Zucchero 65 g
- Amido di mais (maizena) 25 g
- Uova (medie) 1
- Tuorli 1
- Succo di limone ½
- Bacello di vaniglia ½

### PER LA COPERTURA

- Panna acida 100 g
- Frutti di bosco q.b.
- Menta q.b.
- Bacello di vaniglia ½

## Preparazione

Come preparare la New York Cheesecake



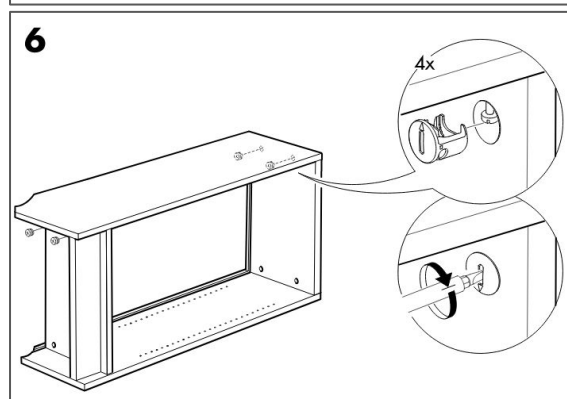
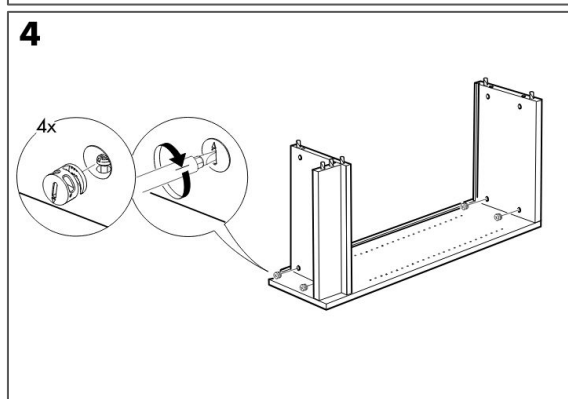
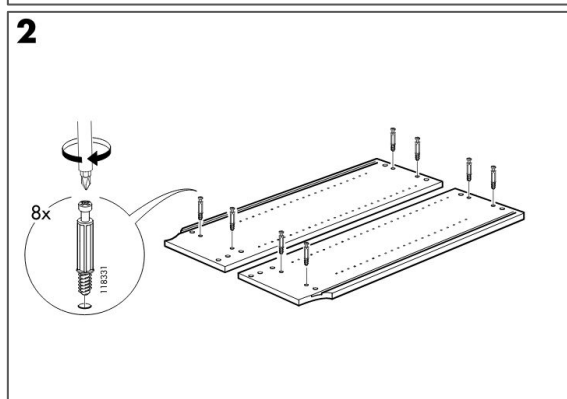
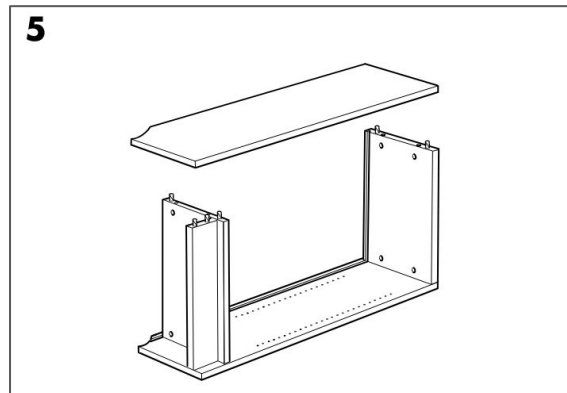
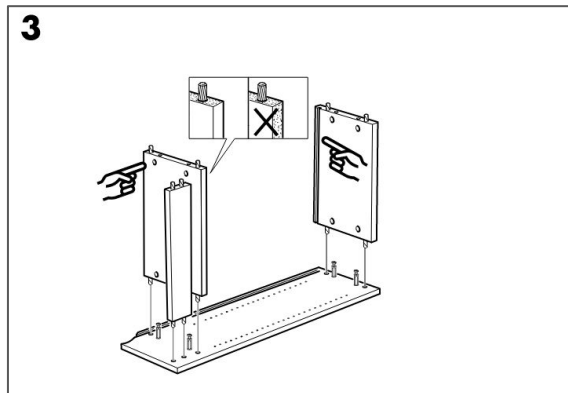
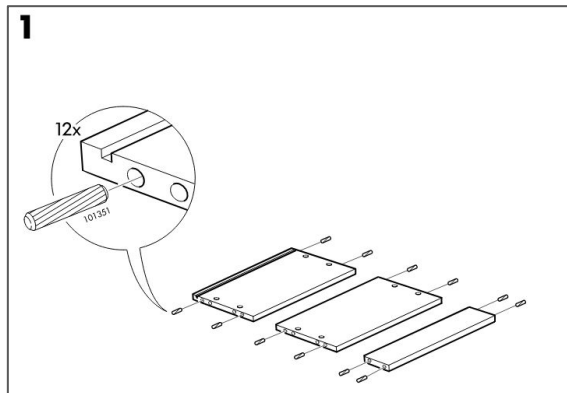
Per preparare la New York cheesecake, per prima cosa fondete il burro e lasciatelo intiepidire; nel frattempo ponete i biscotti in un mixer (1) e frullateli fino a ridurli in polvere (2). Poi trasferiteli in una ciotola e versate il burro (3).



Mescolate con un cucchiaio fino ad uniformare il composto (4); prendete poi uno stampo a cerniera da 22 cm e foderate la base con la carta forno. Ponete metà dei biscotti all'interno e schiacciateli con il dorso del cucchiaio per compattarli (6).

Sorgente: <https://ricette.giallozafferano.it/New-York-Cheesecake.html>

# Situazioni di tutti i giorni 2



# Situazioni di tutti i giorni 3

## A Remark You Made

Joe Zawinul

**Intro**

E♭Maj7 B♭7sus4/E♭

**A**

E♭Maj7 B♭7sus4 E♭ E♭/D Cm7 Cm7/B♭

A♭Maj7 D7 G7 Cm7 A♭/C G/B E♭/B♭ F/A B♭7/A♭

Gm7 Cm7 A♭/C G/B Cm7

Fm7 Fm7/E♭ Fm7/D G7♭9 Cm7 Cm7/B♭

A♭Maj7 G7♭9/A♭ G7♭9 Cm7 E♭Maj7 D♭Maj7 B♭7sus4

E♭Maj7 B♭7sus4 C♯Maj7♯5 E♭/D E♭Maj7 Gm7 Cm7 Cm7/B♭

A Remark You Made (cont.)

A♭Maj7 G7♭9 Cm7 C? Gm7 Cm7 A♭6 F7/A B♭7sus4

**B**

E♭Maj7 E♭/D Cm7 Cm7/B♭ A♭Maj7 A♭/G Fm7 B♭7sus4

**C**

E♭Maj7 B7sus4 B♭7sus4 E♭Maj7 E♭/D Cm7 Cm7/B♭

A♭Maj7 G7♭9 B♭7sus4 A♭Maj7 Gm7 Cm7 A♭Maj7 Fm7

Gm7 Cm7 A♭Maj7 B♭7sus4 G7♭9/B Cm7 A♭Maj7/C

**D** 8 x

Cm7 A♭Maj7/C Gm7 Cm7 D♭Maj7 E♭Maj7

**E** 4 x

Cm7 A♭/C G/B E♭/B♭ F/A B♭7/A♭ Gm7 Cm7 D♭Maj7

**Fim** última vez

# Situazioni di tutti i giorni 4

“Mappa per le discussioni”

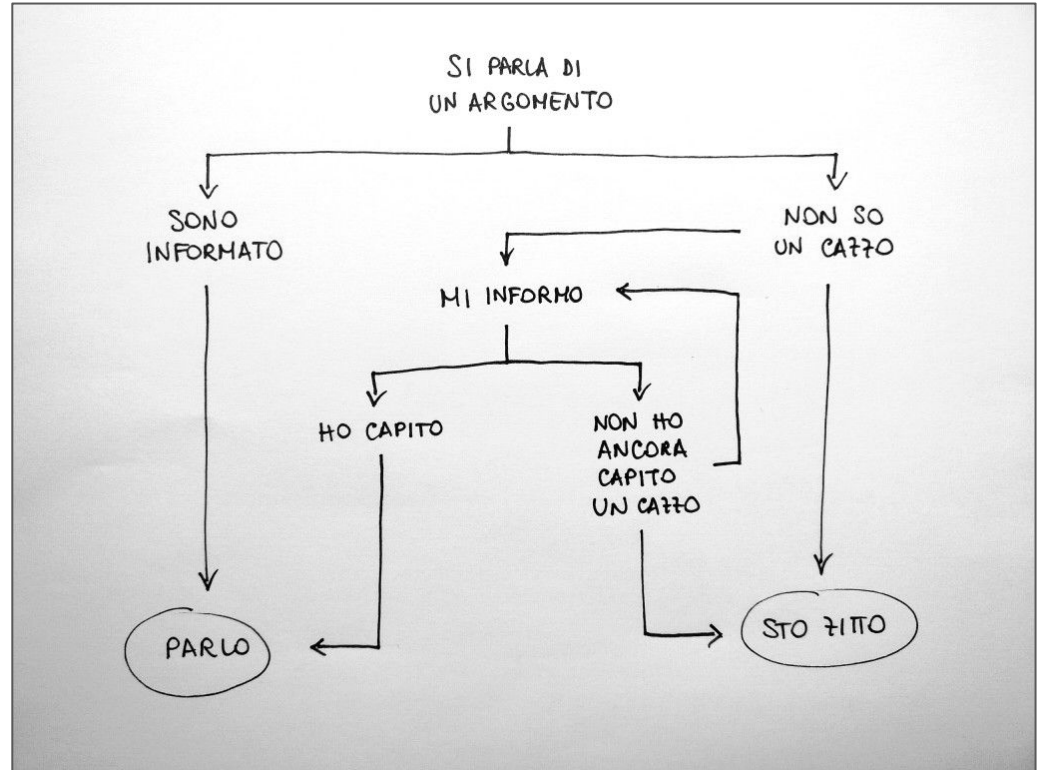
di Federico Cerioni

Pubblicata il 31/8/2016 sul suo profilo Facebook, e poi pubblicata nuovamente sul suo [blog](#)

Postilla dell'autore al post

Facebook originale:

“La mappa è ancora più importante se si sta discutendo di questioni tecniche”



# Cosa hanno in comune queste situazioni?

*Per la base (per una tortiera del diametro di 22 cm)*

- Biscotti Digestive 240 g
- Burro 110 g

**PER LA CREMA**

- Formaggio fresco spalmabile 200 g
- Panna fresca liquida 100 g
- Zucchero 65 g
- Amido di mais (maizena) 25 g
- Moka (medio) 1
- Tuorli 1
- Succo di limone %
- Baccello di vaniglia %

**PER LA COPERTURA**

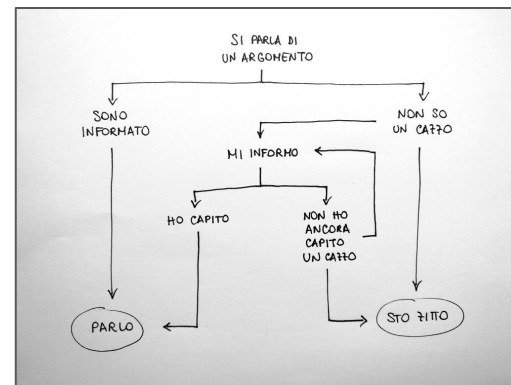
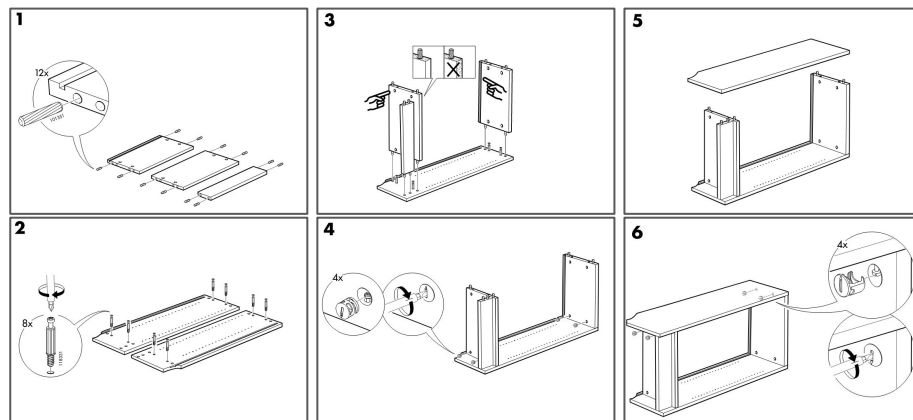
- Panna acida 100 g
- Frutti di bosco q.b.
- Menta q.b.
- Baccello di vaniglia %

**Preparazione**  
Come preparare la New York Cheesecake

Per preparare la New York Cheesecake, per prima cosa fondete il burro e lasciatelo indugiare; nel frattempo ponete i biscotti in un mixer (1) e frullateli fino a ridurli in polvere (2). Poi trasferiteli in una ciotola e versate il burro (3).  
Mescolate con un cucchiaino fino ad uniformare il composto (4); prendete poi uno stampo a cerchiera da 22 cm e foderate la base con la carta forno. Ponete metà dei biscotti all'interno e schiacciateli con il dorso del cucchiaino per compattarli (6).

**A Remark You Made** Joe Zawinul

A Remark You Made (cont.)



# Una definizione

Un **algoritmo** è:

Un'astrazione di una **procedura passo passo** che prende qualcosa come **input** e produce un certo **output**, **scritta in un linguaggio** specifico in modo che le **istruzioni** che definisce possano essere comunicate e comprese da un computer in modo da ottenere qualcosa come conseguenza dell'**elaborazione** di qualche materiale di input

Un programmatore è una persona che **crea algoritmi e li specifica in programmi** usando uno **specifico linguaggio** comprensibile dal computer (elettronico)



# Un passo indietro

Ada Lovelace era una matematica inglese

Partecipò (1833) ad una festa organizzata da Charles Babbage per presentare la Macchina Differenziale, e ne fu così colpita che iniziò una corrispondenza epistolare con lui che durò 27 anni

Fu la traduttrice in inglese del primissimo articolo sulla Macchina Analitica scritto da Luigi Federico Menabrea, e che lei stessa arricchì con un grande numero di annotazioni personali e riflessioni





# Il primo programma

Tra le varie annotazioni che Ada aggiunse al testo, c'era anche una descrizione di come usare la Macchina Analitica per calcolare i numeri di Bernoulli

Questo è riconosciuto come il primo programma della storia dei computer, creato senza avere a disposizione la macchina reale, visto che la Macchina Analitica era soltanto teorica, che di fatto fa di Ada la **prima programmatrice** della storia

“[La Macchina Analitica] potrebbe operare su altre cose oltre ai numeri, se si trovassero oggetti le cui relazioni fondamentali possano essere espresse da quelle della scienza astratta delle operazioni”

Scienza astratta delle operazioni = informatica


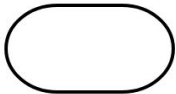

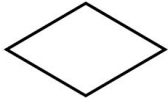

# Che linguaggio usare per definire un algoritmo?

Non esiste un linguaggio standard per descrivere un algoritmo in modo che possa essere immediatamente comprensibile da un qualunque computer

Di solito si usa uno **pseudocodice**, ovvero un linguaggio informale per descrivere i passi principali di un algoritmo ad un umano, che non è direttamente eseguibile da un computer elettronico – anche se i suoi costrutti sono strettamente connessi con quelli tipicamente definiti nei linguaggi di programmazione

Un esempio di pseudocodice: i diagrammi di flusso – li abbiamo informalmente introdotti prima quando abbiamo visto la “Mappa per le discussioni”

# Oggetti grafici di un diagramma di flusso

Oggetto grafico	Nome	Definizione
	Linea di flusso	La freccia è usata per definire l'ordine in cui le operazioni sono eseguite. Il flusso indicato dalla freccia inizia in un terminale di partenza e finisce in un terminale di fine (vedi l'oggetto successivo).
	Terminale	Viene usato per indicare l'inizio e la fine di un algoritmo. Contiene un testo (solitamente o "inizio" o "fine", in italiano) in modo da disambiguare qual è il ruolo del particolare oggetto terminale nel contesto dell'algoritmo.
	Processo	Viene usato per esprimere un'istruzione che è eseguita e che può cambiare lo stato corrente di qualche variabile usata nell'algoritmo. Il testo che contiene descrive l'istruzione da eseguire.
	Decisionale	Permette di esprimere operazioni condizionali, dove una condizione è verificata e, a seconda del valore di alcune variabili usate nell'algoritmo, l'esecuzione continua in un particolare ramo del flusso invece che in un altro. Di solito, questa operazione crea due possibili rami: uno seguito se la condizione è vera, e un altro che viene seguito quando la condizione è falsa.
	Input / Output	Permette di specificare un possibile input o output che viene usato o restituito dall'algoritmo solitamente all'inizio o alla fine della sua esecuzione.

# Il nostro primo algoritmo

Prendi in **input** tre *stringhe*, ovvero due parole e un *riferimento bibliografico* di un articolo pubblicato, e **restituisce**:

- 2 se **entrambe le parole** sono contenute nel riferimento bibliografico
- 1 se **solo una delle parole** è contenuta nel riferimento bibliografico
- 0 **altrimenti**

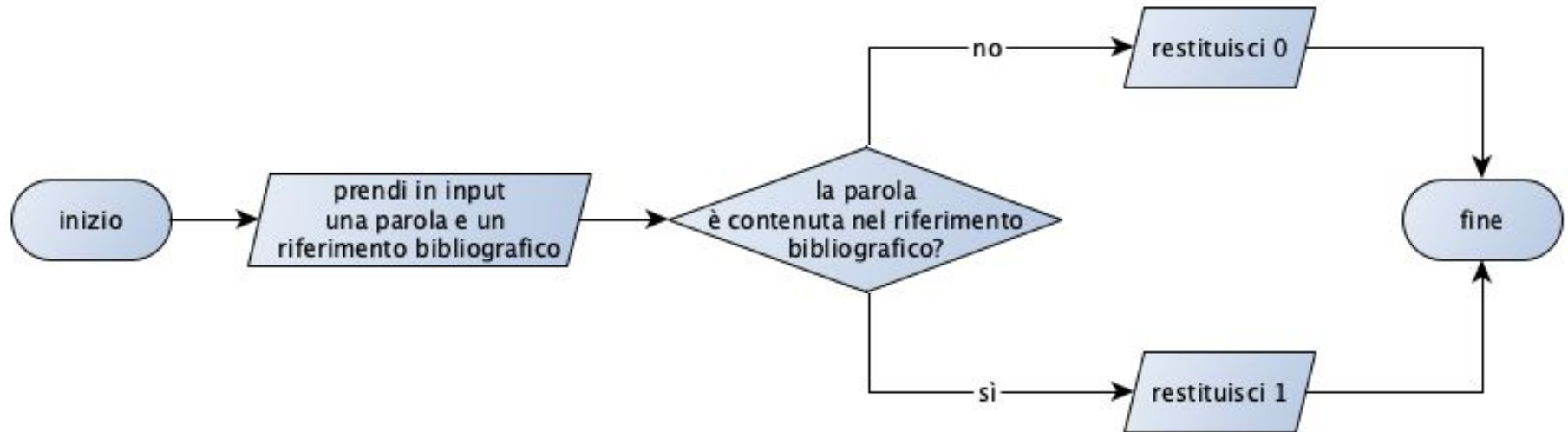
Esempio riferimento bibliografico:

Shannon, C. E. (1948). A Mathematical Theory of Communication. Bell System Technical Journal, 27(3), 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x

Esempio parole: “Theory” e “Journal”, “Conference” e “Bells”, “Logical” e “System”

# Versione parziale

Prendi in input due stringhe, una parola e un riferimento bibliografico, e restituisci 1 se la parola è contenuta nel riferimento bibliografico, 0 altrimenti



# Cosa abbiamo usato

Diverse linee di flusso

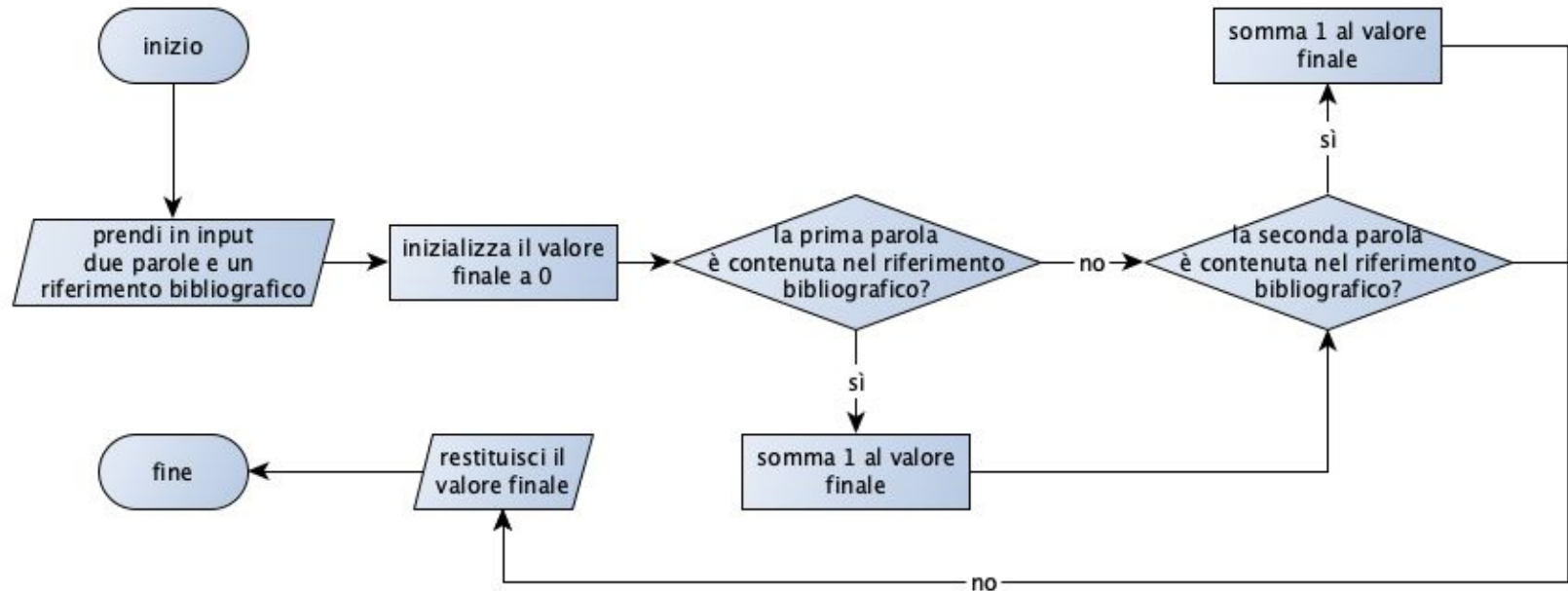
Due oggetti terminali di inizio e fine

Tre oggetti di input / output per acquisire i valori specificati come input e per restituire 0 o 1 dipendentemente da questo input

Un oggetto decisionale dei diagrammi di flusso

# Versione finale

Prendi in input due parole e un riferimento bibliografico e restituisci 2 se entrambe le parole sono contenute nel riferimento bibliografico 1 se solo una delle parole è contenuta nel riferimento bibliografico, 0 altrimenti





# Tre domande chiave del pensiero computazionale

Possiamo usare gli algoritmi per computare qualsiasi cosa vogliamo?

Esiste un limite a quello che possiamo computare?

È possibile definire un **problema computazionale** – ovvero un problema che può essere risolto alitmicamente da un computer – che non può essere risolto da nessun algoritmo?

Che ne pensate?

# Digressione: il paradosso

I paradossi possono essere considerati:

- delle storie divertenti da usare per insegnare
- strumenti potenti che mostrano i limiti di particolari aspetti **formali** di una situazione

Definizione: data una situazione che descrive un particolare problema, qualunque strada che si intraprende per trovare la soluzione del suddetto problema porta ad una contraddizione

# Vediamone uno: la biblioteca di Babele

## Situazione

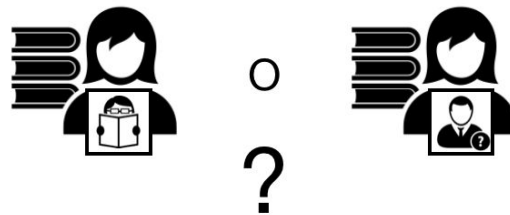


tra queste persone c'è il bibliotecario



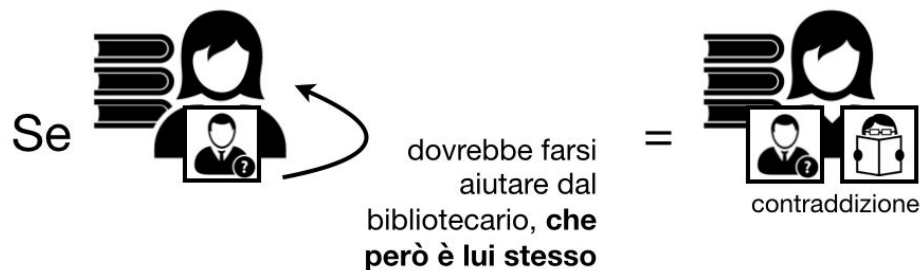
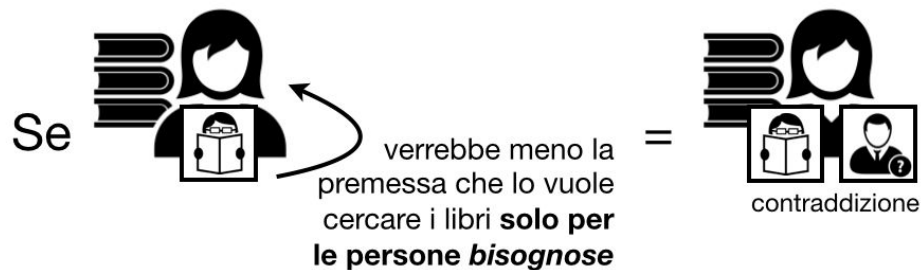
cerca un libro per tutte e sole le persone *bisognose*, ovvero quelle che non sono in grado di cercare quel libro da sole

## Problema



chi cerca i libri al bibliotecario?

# Risoluzione

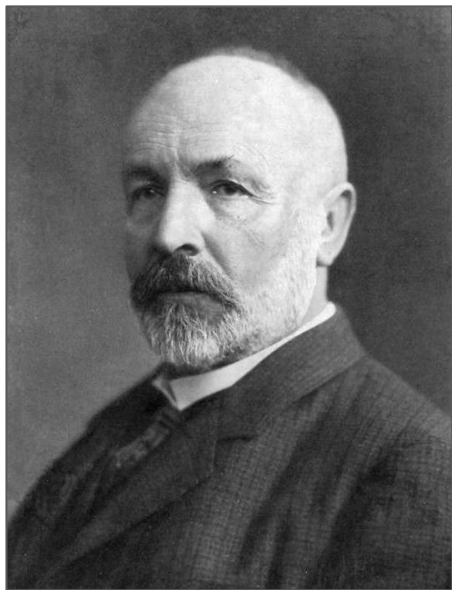


# Perchè i paradossi sono utili

Uno degli approcci più usati per dimostrare che qualcosa non esiste è quello di costruire una situazione in apparenza plausibile che, poi, si rivela paradossale e auto-contraddittoria – in cui, per esempio, **l'esistenza di un algoritmo contraddice se stessa**

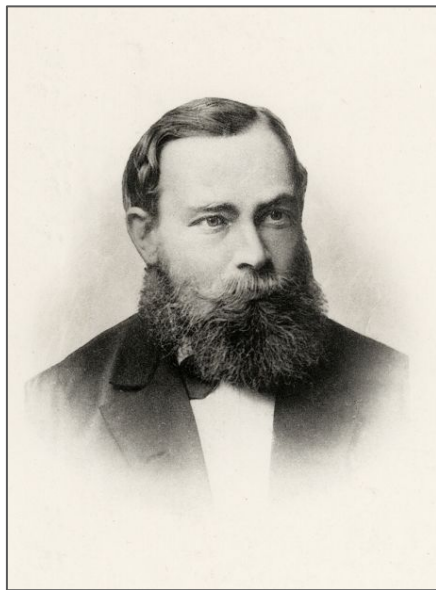
Questo approccio dimostrativo porta il nome di **reductio ad absurdum** (dimostrazione per assurdo): stabilire che una situazione è contraddittoria cercando di derivare un'assurdità dalla sua negazione, in modo da dimostrare che una tesi deve essere accettata perché la sua negazione non può essere difesa e, alla fine, genera un paradosso

# Una storia



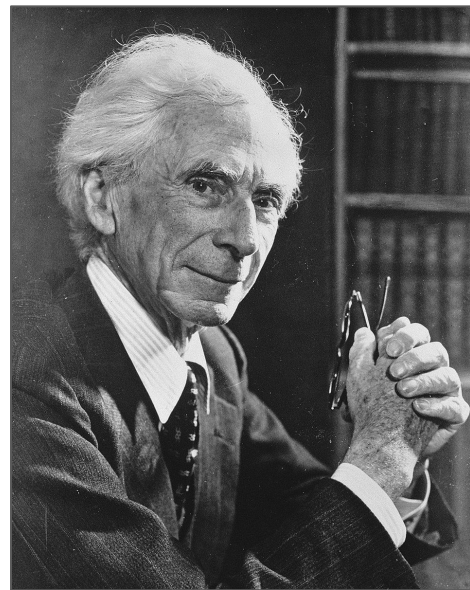
**Georg Cantor**

Crea la teoria degli insiemi (1874-1884), osteggiata da vivo ma compresa come rivoluzionaria in seguito



**Gottlob Frege**

Pubblica il primo volume dei *Principi dell'Aritmetica* (1893) interamente basati sulle idee di Cantor



**Bertrand Russell**

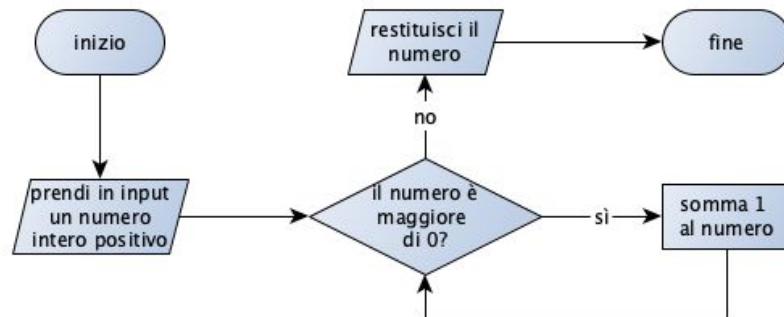
Scopre (1902) un problema nella teoria di Cantor che la invalida, rendendo vana la formalizzazione di Frege

# I problemi aperti della matematica

23 problemi aperti della matematica proposti da David Hilbert 1900

Includono (indirettamente) il **problema della terminazione**: capire se fosse possibile sviluppare un algoritmo che fosse in grado di rispondere se **un altro** algoritmo, specificato come input, terminasse la sua esecuzione o no

È possibile sviluppare un algoritmo che non termina mai la sua esecuzione?

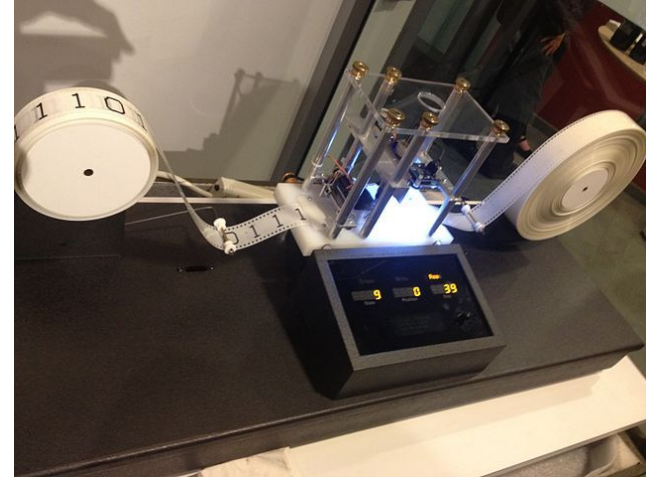
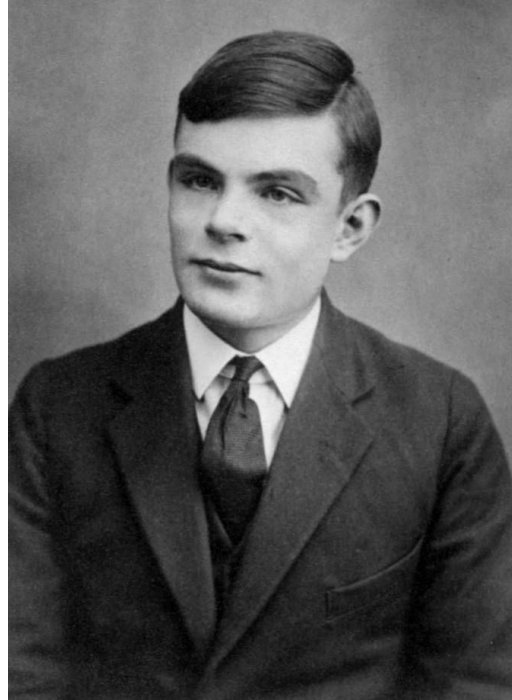




# Alan Turing

Alan Mathison Turing è stato un informatico, padre dell'informatica teorica e dell'intelligenza artificiale

Turing sviluppò la sua macchina teorica (1936) proprio per cercare di rispondere al problema della terminazione di Hilbert



La macchina è in grado di simulare l'esecuzione di qualunque algoritmo realmente implementabile

# Approssimazione della soluzione di Turing

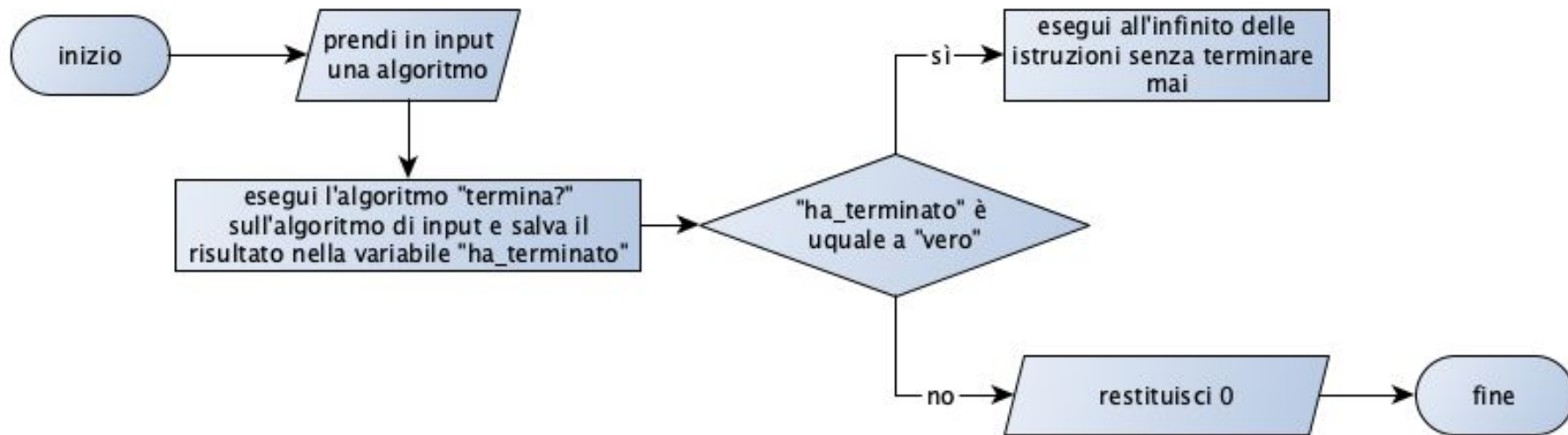
Premessa: supponiamo **sia possibile sviluppare** l'algoritmo "termina?", che prende in **input un certo algoritmo** e restituisce "**vero**" nel caso in cui l'algoritmo specificato come **input termina**, mentre restituisce "**falso**" **in caso contrario**

NB: è soltanto un algoritmo ipotetico, stiamo supponendo che possiamo svilupparlo in qualche modo, senza mostrare come farlo davvero

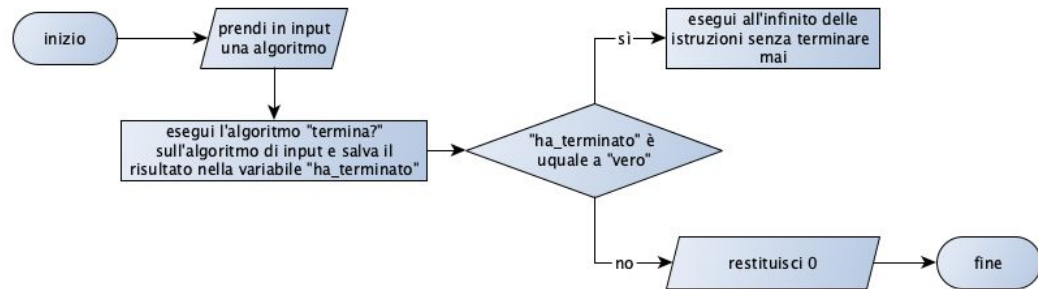
Usiamo questo algoritmo per crearne un altro

# Nuovo algoritmo

Nuovo algoritmo: prendi in input un algoritmo e restituisci 0 se l'algoritmo in input non termina, mentre non termina in caso contrario



# E se usiamo il nuovo algoritmo come suo input?



Caso 1: se l'algoritmo "termina?" afferma che il nuovo algoritmo **termina**, conseguentemente (per come è definito) il nuovo algoritmo **non termina** l'esecuzione

Caso 2: se l'algoritmo "termina?" afferma che il nuovo algoritmo **non termina**, e conseguentemente (per come è definito) il nuovo algoritmo restituisce 0 e **termina** l'esecuzione

Paradosso: l'algoritmo che verifica se un altro termina **non può esistere**

# I limiti della computazione

Questo risultato ha avuto un effetto dirompente sulla percezione delle abilità computazionali che un computer può avere

La macchina di Turing e le relative analisi effettuate su di essa hanno imposto dei **limiti** chiarissimi a quello che possiamo calcolare, e hanno permesso di dimostrare che determinati problemi computazionali interessanti, come quello della terminazione, non possono essere risolti **da nessun approccio algoritmico**

Tutto questo è stato possibile solo grazie all'applicazione di un pensiero computazionale esclusivamente astratto, considerando che la macchina di Turing è solo uno strumento prettamente teorico

# Fine

## Algoritmi e computabilità

Informatica di base – a.a. 2022/2023

Silvio Peroni

[0000-0003-0530-4305](mailto:silvio.peroni@unibo.it)

Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia

[silvio.peroni@unibo.it](mailto:silvio.peroni@unibo.it) – [@essepuntato](https://www.unibo.it/sitoweb/silvio.peroni) – <https://www.unibo.it/sitoweb/silvio.peroni/>



Quest'opera è distribuita con [Licenza Creative Commons Attribuzione 4.0 Internazionale](https://creativecommons.org/licenses/by/4.0/)



DIPARTIMENTO DI FILOLOGIA CLASSICA E ITALIANISTICA