

PROGRAMMAZIONE I

A.A. 2018/2019

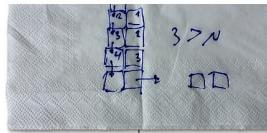
COMPUTATIONAL THINKING

- ☞ Formalizzato da una psicologa Americana Jeanette Wing in un articolo sulla rivista ACM nel 2008
 - ✓ <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2696102/>
- ☞ Il **pensiero computazionale** è l'insieme dei processi mentali coinvolti nella formulazione di un problema e della sua soluzione in modo tale che un umano o una macchina possa effettivamente eseguire.
- ☞ Il pensiero computazionale è un processo iterativo basato su tre fasi:
 - ✓ Formulazione del problema (astrazione);
 - ✓ Espressione della soluzione (automazione);
 - ✓ Esecuzione della soluzione e valutazione della stessa (analisi).

Computational thinking steps

Abstraction

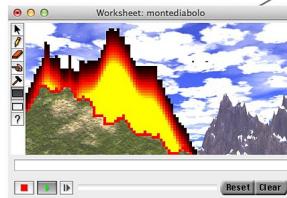
Problem Formulation



"how does a mudslide work?"

Analysis

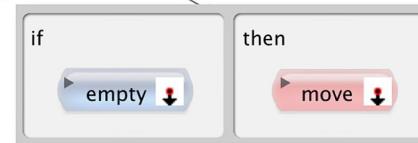
Solution Execution and Evaluation



visualize the consequence of thinking

Automation

Solution Expression



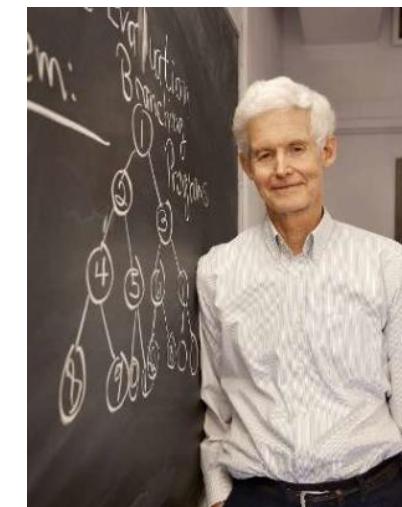
build simple model of gravity

Algorithmi e coding

- ✓ Regole logiche da impiegare nella risoluzione di un problema e la sequenza in cui devono essere applicate
- ✓ Esecuzione dei passi della sequenza sopra su un particolare dispositivo di calcolo

STORIA DELLA SCIENZA DELLA COMPUTAZIONE

- ⌚ [Fine '800 inizi '900] Studi di insiemistica e logica matematica (Cantor, Hilbert, Gödel)
- ⌚ [1936] Definizione formale di algoritmo e di funzione calcolabile, Church e Turing in contemporanea
- ⌚ [1971] Individuazione di problemi intrattabili, Cook e Levin in contemporanea



GÖDEL E L'ESISTENZA DI DIO



Assioma 1. $P(\varphi) \cdot P(\psi) \supset P(\varphi \cdot \psi)$.

Assioma 2. $P(\varphi) \vee P(\sim \varphi)$.

Definizione 1. $G(x) \equiv (\varphi)[P(\varphi) \supset \varphi(x)]$. (Dio)

Definizione 2. $\varphi \text{Ess. } x \equiv (\psi)[\psi(x) \supset N(y)[\varphi(y) \supset \psi(y)]]$. (Essenza di x)

$$p \supset_N q = N(p \supset q).$$

Necessità

Assioma 3. $\begin{array}{l} P(\varphi) \supset NP(\varphi) \\ \sim P(\varphi) \supset N \sim P(\varphi) \end{array}$

Teorema. $G(x) \supset G \text{ Ess. } x$.

Definizione. $E(x) \equiv (\varphi)[\varphi \text{Ess. } x \supset N(\exists x)\varphi(x)]$. (Esistenza necessaria)

Assioma 4. $P(E)$.

Teorema. $G(x) \supset N(\exists y)G(y)$,

quindi $(\exists x)G(x) \supset N(\exists y)G(y)$;

quindi $M(\exists x)G(x) \supset M N(\exists y)G(y)$. (M = possibilità)

$M(\exists x)G(x) \supset N(\exists y)G(y)$.

$M(\exists x)G(x)$ significa che il sistema di tutte le proprietà positive è compatibile. Ciò è vero grazie a:

Assioma 5. $P(\varphi) \cdot \varphi \supset_N \psi : \supset P(\psi)$, che implica

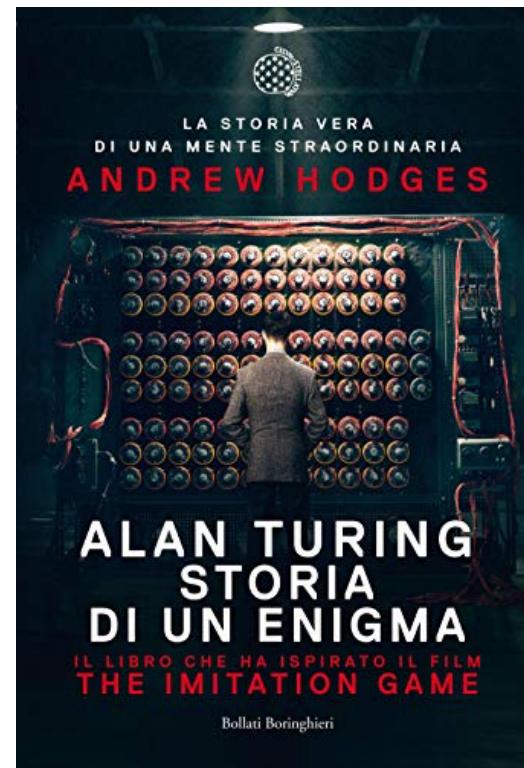
$$\begin{cases} x = x & \text{è positivo} \\ x \neq x & \text{è negativo.} \end{cases}$$

ALAN TURING



- ⌚ Matematico e crittoanalista
- ⌚ Collaborando alla “Bomba” è riuscito a rompere la cifratura effettuata dai Nazisti durante la 2mondiale
 - ✓ Alcuni analisti pensano che grazie a lui la guerra sia durata due anni meno
- ⌚ Inventore della Macchina di Turing
- ⌚ Padre dell’Intelligenza Artificiale
 - ✓ Test di Turing (nessun computer lo ha passato fino ad oggi)
- ⌚ https://it.wikipedia.org/wiki/Premio_Turing

ALAN TURING



I RISULTATI SONO

- ⌚ Non tutte le funzioni matematiche sono calcolabili con un algoritmo.
 - ✓ Le funzioni matematiche sono più degli algoritmi

- ⌚ Per alcuni problemi (intrattabili), il numero di operazioni richieste aumenta esponenzialmente rispetto alla dimensione dei dati trattati
 - ✓ Per esempio rispetto alla lunghezza di un array
 - ✓ Non sappiamo ancora se qualche algoritmo più efficiente possa essere ideato (si pensa di no)

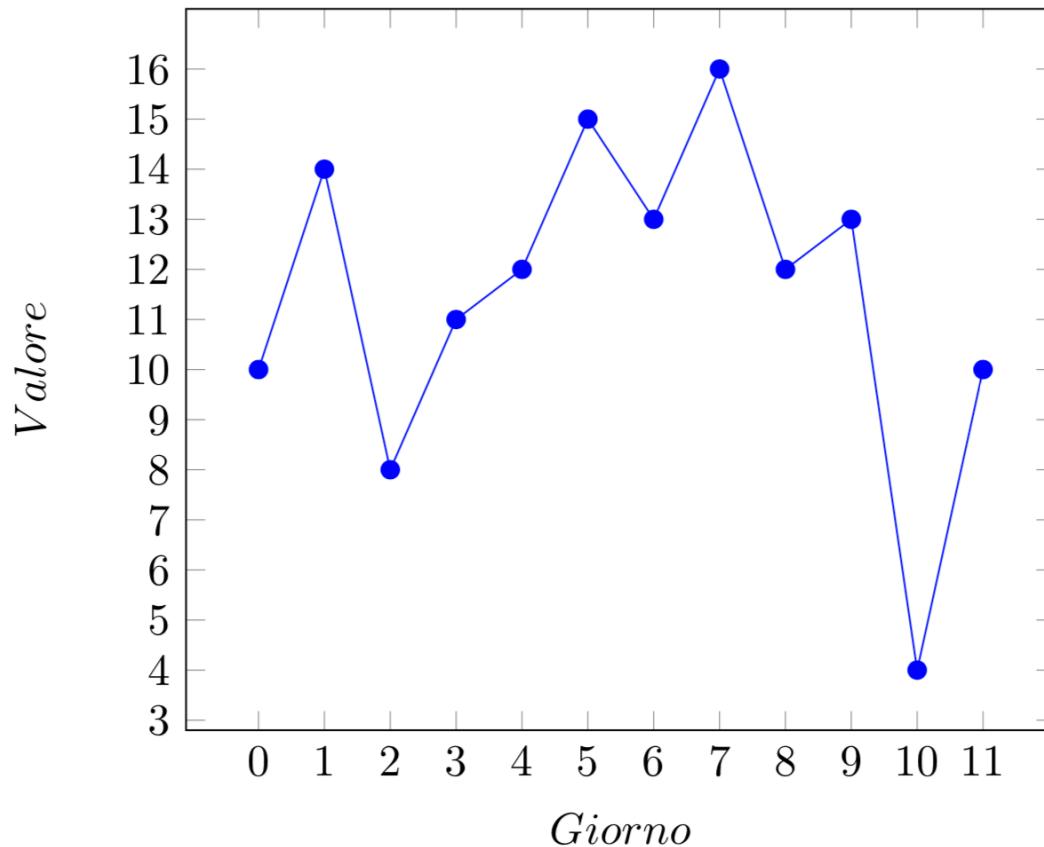
UN ESEMPIO VICINO A NOI

- ⌚ Un algoritmo (programma) che prende in input un programma ed un input e stabilisce se il programma termina su tale input NON ESISTE (non è calcolabile).
- ✓ Halting problem

BACK TO ESONERO

- ② Computabilità ma anche complessità (ritorno all'esercizio dell'esonero)
- ② L'esercizio dell'esonero può essere risolto analizzando l'array, scandendolo con dei for
- ② Per pseudocodice, pseudocodifica, pseudolinguaggio o linguaggio di progettazione si intende un linguaggio il cui scopo è la rappresentazione di algoritmi in alternativa al classico diagramma di flusso e non soggetto a molte limitazioni intrinseche di quest'ultimo tipo di rappresentazione

INTERVALLO DI SOMMA MASSIMA



+4	-6	+3	+1	+3	-2	+3	-4	+1	-9	+6
----	----	----	----	----	----	----	----	----	----	----

ALGORITMO CUBICO

programma CUBICO (D)

// Il vettore $D[1 : n]$ consiste di n numeri positivi e negativi

1. $MaxSomma \leftarrow -\infty; a \leftarrow 1; v \leftarrow 0;$
2. **for** $i = 1$ **to** n
3. **for** $j = i$ **to** n
4. $Tmp \leftarrow 0$; // Tmp è un valore temporaneo
5. **for** $k = i$ **to** j
6. $Tmp \leftarrow Tmp + D[k];$
7. **if** $Tmp > MaxSomma$
8. $MaxSomma \leftarrow Tmp; a \leftarrow i; v \leftarrow j;$
9. **print** “Il guadagno massimo è” $MaxSomma;$
10. **print** “Esso si realizza nell’intervallo di giorni” $[a, v];$

ALGORITMO QUADRATICO

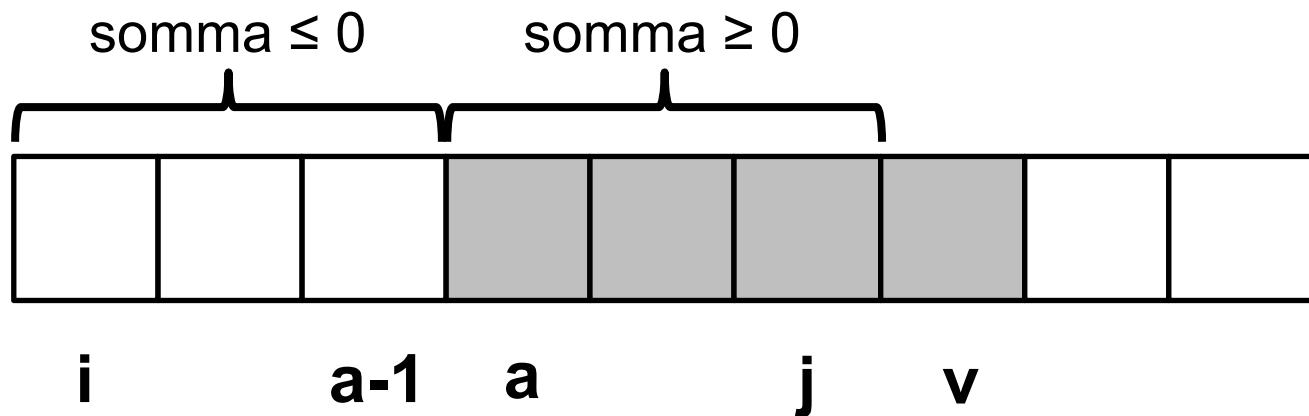
programma QUADRATICO (D)

// Il vettore $D[1 : n]$ consiste di n numeri positivi e negativi

1. $MaxSomma \leftarrow -\infty; a \leftarrow 1; v \leftarrow 0;$
2. **for** $i = 1$ **to** n
3. $Tmp \leftarrow 0;$
4. **for** $j = i$ **to** n
5. $Tmp \leftarrow Tmp + D[j];$
6. **if** $Tmp > MaxSomma$
7. $MaxSomma \leftarrow Tmp; a \leftarrow i - 1; v \leftarrow j;$
8. **print** “Il guadagno massimo è” $MaxSomma;$
9. **print** “Esso si realizza nell’intervallo di giorni” $[a, v];$

STUDIANDO MEGLIO IL PROBLEMA

Da a a v è l'intervallo di somma massima



Proposizione 1: Ogni porzione che termina immediatamente prima della porzione di somma massima (da i a $a-1$), ha somma negativa (dimostrazione per assurdo).

Proposizione 2: Ogni porzione che inizia dove inizia la porzione di somma massima ed è inclusa in essa (da a a j con $a \leq j \leq v$), ha somma positiva (dimostrazione per assurdo).

IN PRATICA

- ② Sommo passo passo tutti gli elementi dell'array (quello con le differenze tra giorno successivo e giorno precedente) dall'indice 0 in poi
- ② Appena trovo che la somma fino ad un indice ($a-1$) è minore di zero, sono sicuro di poter scartare tutto l'intervallo fino a $a-1$. L'intervallo massimo potrà esistere da questo punto in poi
- ② a e v nella prossima slide vengono aggiornati con l'intervallo di costo massimo, in base alle proprietà della slide precedente

ALGORITMO LINEARE

programma LINEARE (D)

// Il vettore D consiste di n numeri positivi e negativi

1. $MaxSomma \leftarrow -\infty; v \leftarrow 0; Tmp \leftarrow 0;$
2. $a \leftarrow 1; atmp \leftarrow 1;$
3. **for** $i = 1$ **to** n
4. $Tmp \leftarrow Tmp + D[i];$
5. **if** $Tmp > MaxSomma$
6. $MaxSomma \leftarrow Tmp; v \leftarrow i; a \leftarrow atmp;$
7. **if** $Tmp < 0$
8. $Tmp \leftarrow 0; atmp \leftarrow i + 1;$
9. **print** "Il guadagno massimo è" $MaxSomma;$
10. **print** "Esso si realizza nell'intervallo di giorni" $[a, v];$

COMPLESSITÀ (UN'INTRODUZIONE)

- ☞ Data la lunghezza n di un array
 - ☞ L'algoritmo cubico esegue $O(n^3)$ passi. Almeno $n^3/32$
 - ☞ L'algoritmo quadratico esegue $O(n^2)$ passi
-
- ☞ Se il nostro computer esegue un miliardo di operazioni al secondo,
 - ✓ L'algoritmo cubico impiega per processare un array di un milione di elementi: $(10^6)^3 \times 10^{-9} = 10^9$ (piu di 30 anni)
 - ✓ L'algoritmo quadratico ci mette $(10^6)^2 \times 10^{-9} = 10^3$ (16 minuti)
 - ✓ L'algoritmo lineare $10^6 \times 10^{-9} = 10^{-3}$ (un millesimo di secondo)

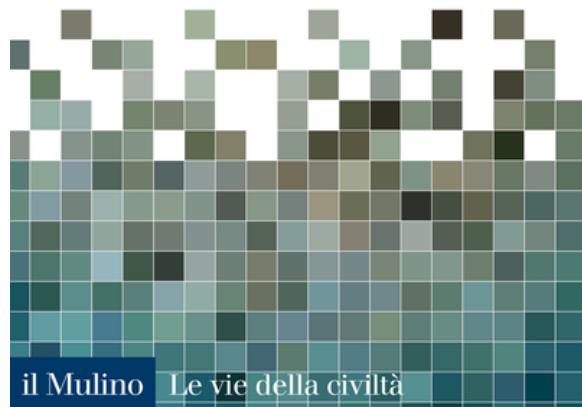
E NEL FUTURO?

- ⌚ È meglio sviluppare un algoritmo più efficiente o comprare un computer più potente?
- ⌚ Quanti elementi processo in un tempo T?
 - ✓ $\sqrt[3]{T}$, \sqrt{T} e T
- ⌚ E se compro un computer più potente che mi fa k operazioni in un tempo T?
 - ✓ $\sqrt[3]{kT}$, \sqrt{kT} e kT che equivale a
 - ✓ $\sqrt[3]{k} \times \sqrt[3]{T}$, $\sqrt{k} \times \sqrt{T}$ e kT
- ⌚ Comprare un computer più potente rende più veloce l'algoritmo già più efficiente

LETTURA CONSIGLIATA

Paolo Ferragina
Fabrizio Luccio

**Il pensiero
computazionale**
Dagli algoritmi al coding



Creare l'algoritmo, pensare al problema, è più importante che scrivere il programma in un linguaggio di programmazione.

Un informatico risolve i problemi.

