Elementi di Teoria della Computazione

(ETC)

A.A. 2016/17

MATRICOLE DISPARI

Docente: Prof. Luisa Gargano

BENVENUTI!

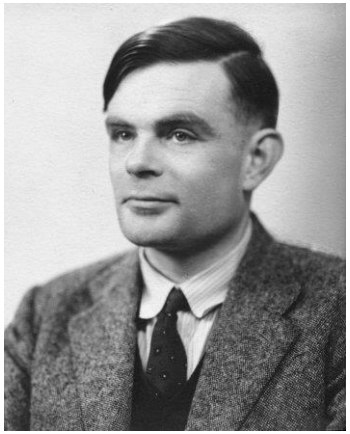
Finalità: Fornire gli elementi di base delle teorie che sono di fondamento all'informatica

1. Computabilità
2. Complessità

Computazione

Obiettivo:

- **Formulare una *teoria* a partire dall'idea della *computazione***



Alan Turing

Cosa è la "Computazione"?

Treccani

- ***Computazione:*** *Il computare e il modo con cui si computa; calcolo.*

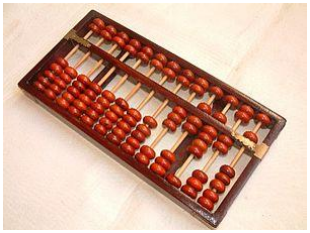
- ***Computare:*** *Calcolare, fare il conto di qualche cosa: c. il tempo necessario; metodo di c. gli anni;*

Mezzi di "Computazione"

- **Carta e Penna**

$$\begin{array}{r} 237+ \\ 84= \\ \hline 321 \end{array}$$

- **Abaco**



- **….**
- **Calcolatori/programmi**
- **…**

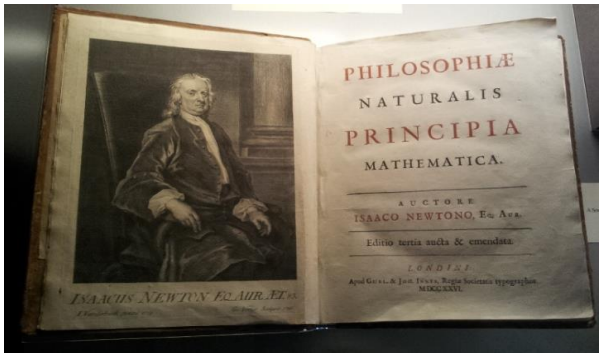
Cosa è la "Computazione"?

Per noi la computazione sarà

processare l'informazione
mediante l'applicazione illimitata di un
insieme finito di operazioni o regole

Es. Linguaggio macchina

Teoria



Che cosa si vuole da una teoria?

- Precisione,
- Generalità

"Teoria della Computazione"?

- **Generalità**
  - Possiamo definire la computazione senza fa riferimento ad un calcolatore attuale?
  - Possiamo definire la computazione indipendentemente dai limiti odierni della scienza (ingegneria, fisica,…)?

"Teoria della Computazione"?

- **Generalità**
  - Possiamo definire la computazione senza fa riferimento ad un calcolatore attuale?
  - Possiamo definire la computazione indipendentemente dai limiti odierni della scienza (ingegneria, fisica,…)?

- **Precisione**
  - Possiamo definire formalmente (*matematicamente*) un calcolatore?
  - Possiamo dimostrare teoremi circa ciò che può o non può essere computato?

Avendo a disposizione risorse (memoria, tempo,…) sufficienti

- un *calcolatore* può risolvere qualsiasi problema?
- oppure esistono limiti fondamentali a ciò che si può computare?

Computabilità: *Quali problemi possono essere computati?*
(con qualsiasi macchina, linguaggio, …)

Esempi di problemi computazionali

- Problemi numerici
  - Data una stringa binaria, il numero di 1 è maggiore del numero di 0?
  - Dati due numeri x e y, calcola x+y
  - Dato un intero, risulta x primo?
- Problemi riguardanti programmi (es. in C)
  - Data una sequenza di caratteri ASCII, rispetta la sintassi del C?
  - Dato un programma in C, esiste un input che lo manda in loop?

Computabilità: *Quali problemi possono essere computati?*
(con qualsiasi macchina, linguaggio, …)

Esempi di problemi computazionali in matematica

- Equazioni Diofantine: data un'equazione con una o più incognite e coefficienti interi (es $x^2+3xyz-37z^3-5=0$), essa ammette una soluzione intera?

- Problema del commesso viaggiatore: data una rete di città, connesse tramite delle strade, trovare il percorso di minore lunghezza che un commesso viaggiatore deve seguire per visitare tutte le città una e una sola volta per poi tornare alla città di partenza

Computabilità: *Quali problemi possono essere computati?*
(con qualsiasi macchina, linguaggio, …)

Macchine a stati finiti/Automi:

Quali problemi possiamo risolvere con ***memoria costante****?*

Macchina a stati finiti

Macchina a stati finiti

Distributore di bibite/snack a 50c

Accetta monete da 10c e da 20c non da resto, rifiuta moneta troppo grande

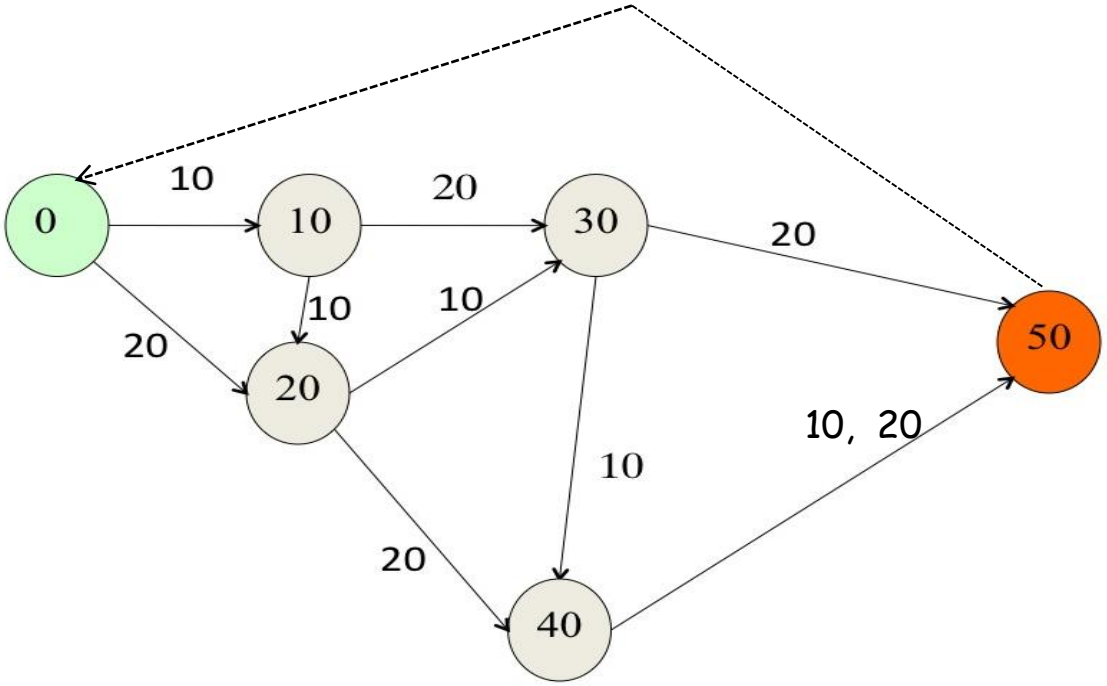
Distributore di bibite/snack a 50c

Accetta monete da 10c e da 20c
non da resto, rifiuta moneta troppo grande

E' (semplice) **macchina a stati finiti** che opera in accordo all'input (monete)

Distributore di bibite/snack a 50c

Accetta monete da 10c e da 20c
non da resto, rifiuta moneta troppo grande

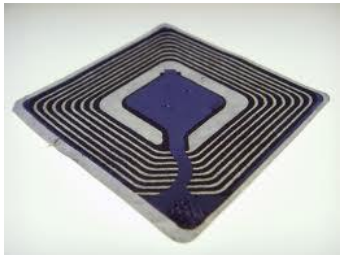


E' (semplice) **macchina a stati finiti** che opera in accordo all'input (monete)

Macchina a stati finiti o Automa finito

Astrazione ⇔ modello di calcolo più semplice

- Chip



.Componente elettronico su cui è presente un circuito integrato, cioè un circuito elettronico miniaturizzato
Parte di molti apparecchi elettromeccanici

  - Analizzatori lessicali e sintattici di compilatori,
    - Wikipedia: Un **compilatore** è un programma che traduce una serie di istruzioni scritte in un determinato linguaggio di programmazione (codice sorgente) in istruzioni di un altro linguaggio (codice oggetto).

- ...

Computabilità: *Quali problemi possono essere computati?*
(con qualsiasi macchina, linguaggio, …)

Esempi di problemi computazionali

- Problemi numerici
  - Data una stringa binaria, il numero di 1 è maggiore del numero di 0?
  - Dati due numeri x e y, calcola x+y
  - Dato un intero, risulta x primo?
- Problemi riguardanti programmi (es. in C)
  - Data una sequenza di caratteri ASCII, rispetta la sintassi del C?
  - Dato un programma in C, esiste un input che lo manda in loop?

Computabilità: *Quali problemi possono essere computati?*

Non tutti!!!

Esempi di problemi computazionali

- Problemi numerici
  - Data una stringa binaria, il numero di 1 è maggiore del numero di 0?
  - Dati due numeri x e y, calcola x+y
  - Dato un intero, risulta x primo?
- Problemi riguardanti programmi (es. in C)
  - Data una sequenza di caratteri ASCII, rispetta la sintassi del C?
  - Dato un programma in C, esiste un input che lo manda in loop?

Computabilità: *Quali problemi possono essere computati?*

Non tutti!!!

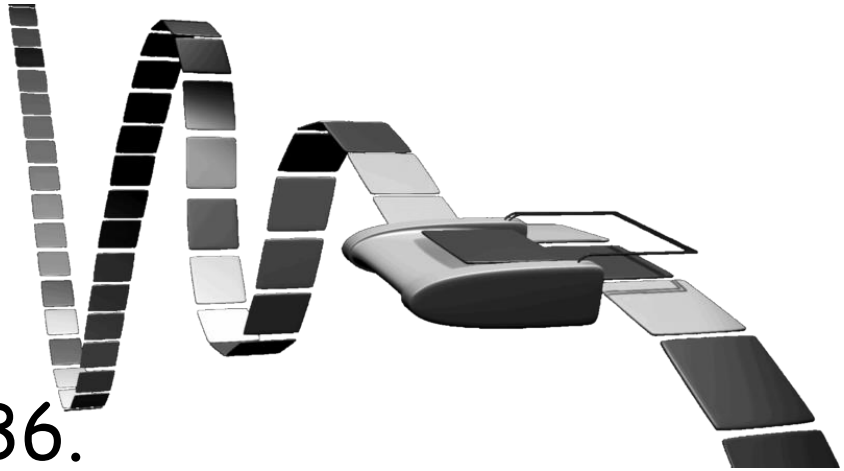
Es. Dato un qualsiasi programma in C, possiamo stabiliure se termina su ogni input? Risposta: NO

```
input n;
while (n !=1)
{
if (n is even)
n := n/2;
else
n := 3*n+1;
}
```

Termina per ogni n>1? 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Computabilità: *Modello di computazione indipendente dalla tecnologia presente?*

Macchine di Turing



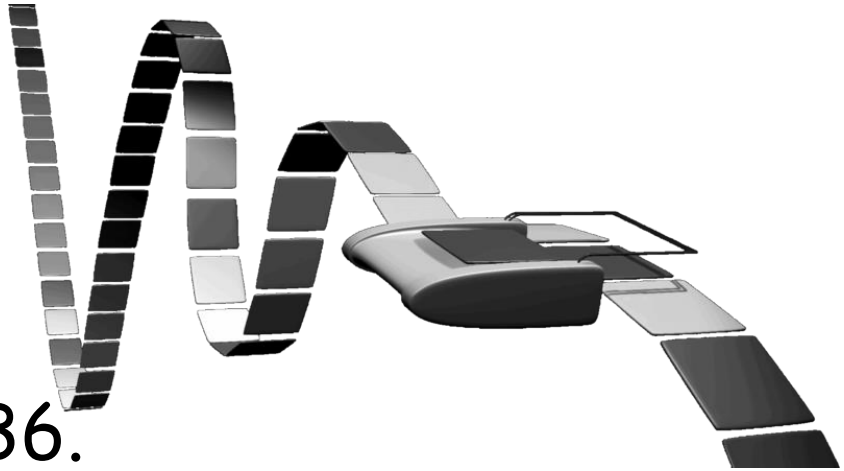
Ideate da Alan Turing nel 1936.

Modello di calcolatore più semplice:

- macchina a stati finiti

Computabilità: *Modello di computazione indipendente dalla tecnologia presente?*

Macchine di Turing



Ideate da Alan Turing nel 1936.

Modello di calcolatore più semplice:

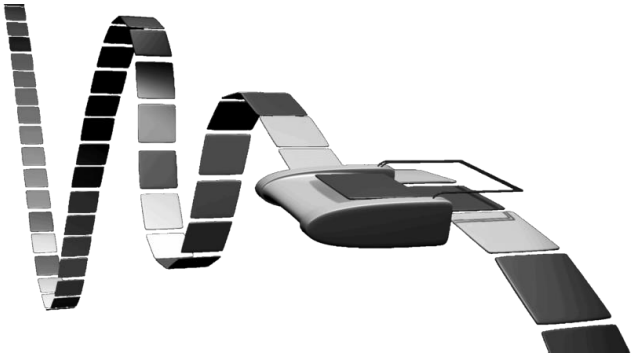
- macchina a stati finiti
- **Nastro (lettura e scrittura) ⇔ memoria, processore**

Macchine di Turing

Ideate da Alan Turing nel 1936.

Modello di calcolatore più semplice:

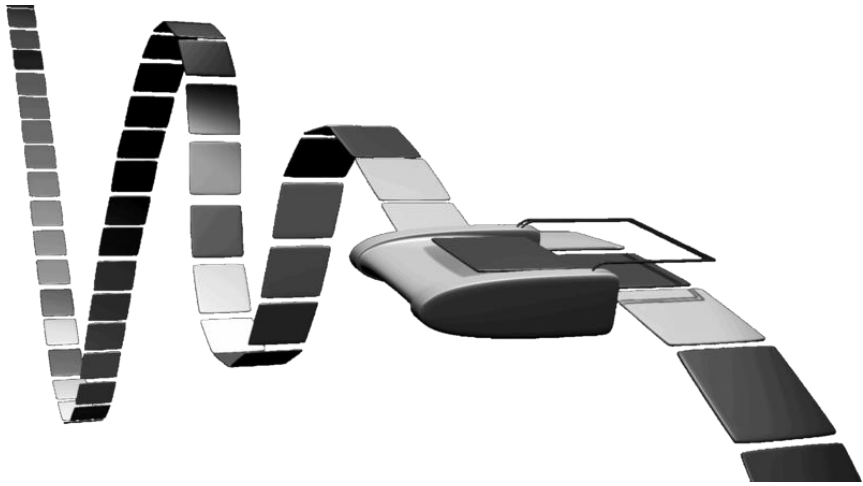
- macchina a stati finiti
- **Nastro ⟺ memoria, processore**



Scopo:

fornire una teoria della computazione

formalizzare in maniera esatta (matematica) il concetto di computazione (indipendentemente dalla "potenza di calcolo")

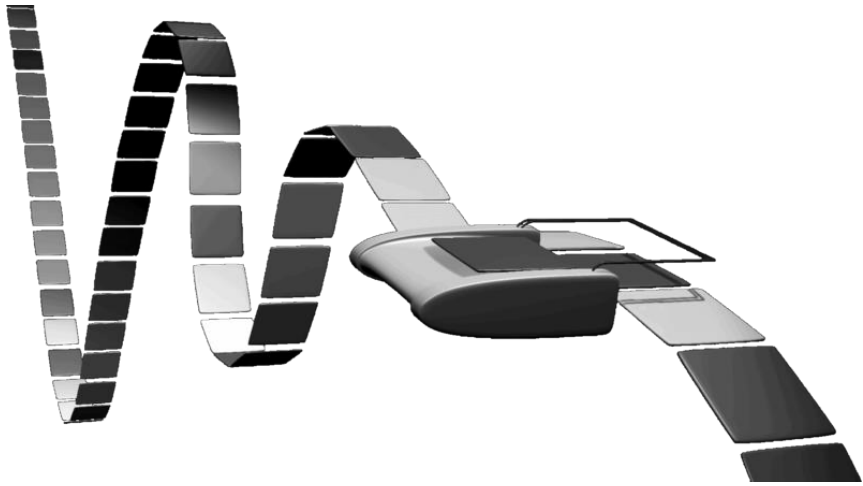


- **Macchine di Turing:**

  Concetto di Computabilità indipendente dalla tecnologia

- **Tesi di Church-Turing:**

  Equivalenza tra programmi e Macchine di Turing



- **Macchine di Turing:**

  Concetto di Computabilità indipendente dalla tecnologia

- **Tesi di Church-Turing:**

  Equivalenza tra programmi e Macchine di Turing

- **Limiti delle macchine di Turing (e della computazione):**

  Problemi non computabili

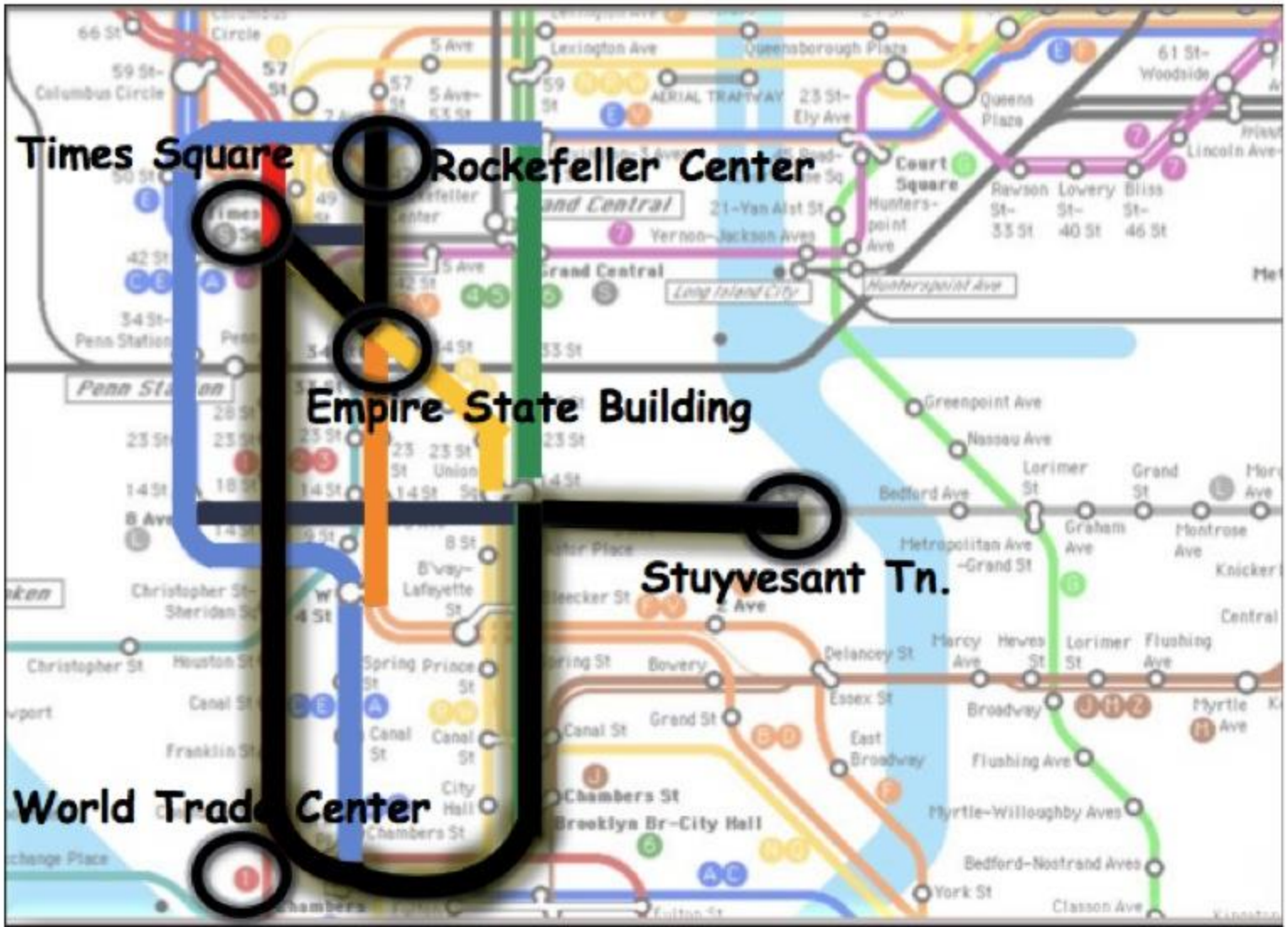
Computabilità: *Cosa può essere computato?*
Cosa non può esserlo?
(con qualsiasi macchina, linguaggio, …)
Dove si trova il confine?

Complessità: *Quali sono le le risorse minime necessarie*
(es. tempo di calcolo e memoria)
per la risoluzione di un problema?

Lista di luoghi con associato l'interesse a visitare il luogo (es. voto da 0 a 100)

Vogliamo ordinare i luoghi in ordine di interesse

Facile!



Vogliamo pianificare un giro turistico che visita tutti i posti di interesse esattamente 1 volta

Usando i collegamenti esistenti (metro, bus). Facile?

Computabilità: *Quali problemi possono essere computati?*
(con qualsiasi macchina, linguaggio, …)

Esempi di problemi computazionali in matematica

- Equazioni Diofantine: data un'equazione con una o più incognite e coefficienti interi (es $x^2+3xyz-37z^3-5=0$), essa ammette una soluzione intera?

- **Problema del commesso viaggiatore:** data una rete di città, connesse tramite delle strade, trovare il percorso di minore lunghezza che un commesso viaggiatore deve seguire per visitare tutte le città una e una sola volta per poi tornare alla città di partenza

Algoritmo semplice (Backtrack)

Quanto tempo occorre?

Siti	Passi
N	N!
5	120
15	1307674368000
100	$\approx 9 \cdot 10^{157}$

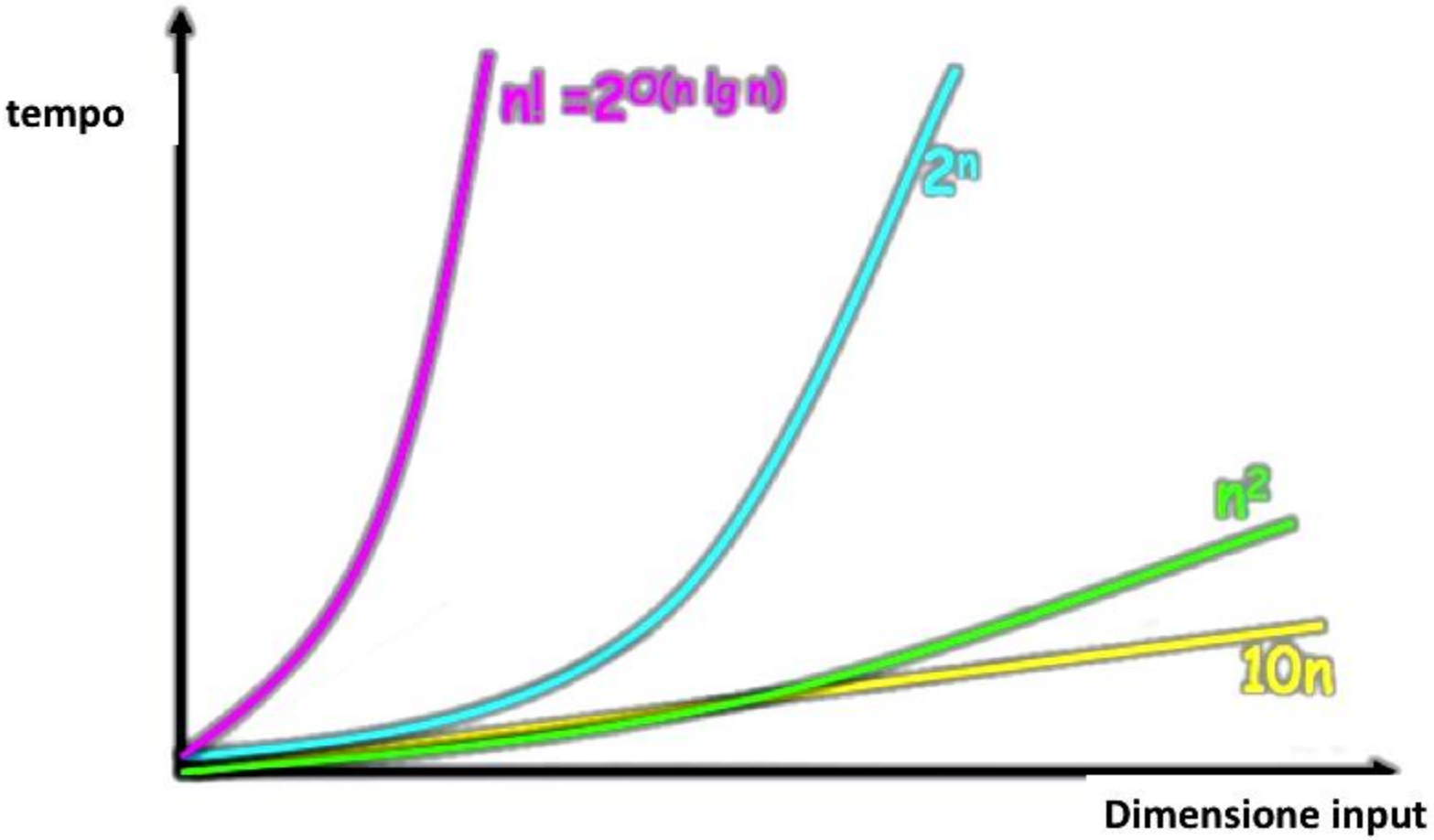
4 anni su computer che può valutare 10.000 opzioni al secondo

Quanto tempo occorre?

Siti	Passi
N	N!
5	120
15	1307674368000
100	$\approx 9 \cdot 10^{157}$!!!!

Quasi 6 mesi su calcolatore 100 volte più veloce (1.000.000 di operazioni al sec.)

Tasso di crescita: una prima classificazione



Algoritmi utili in pratica

Algoritmi efficienti: Utilizzano un tempo **polinomiale** su **tutti** gli input

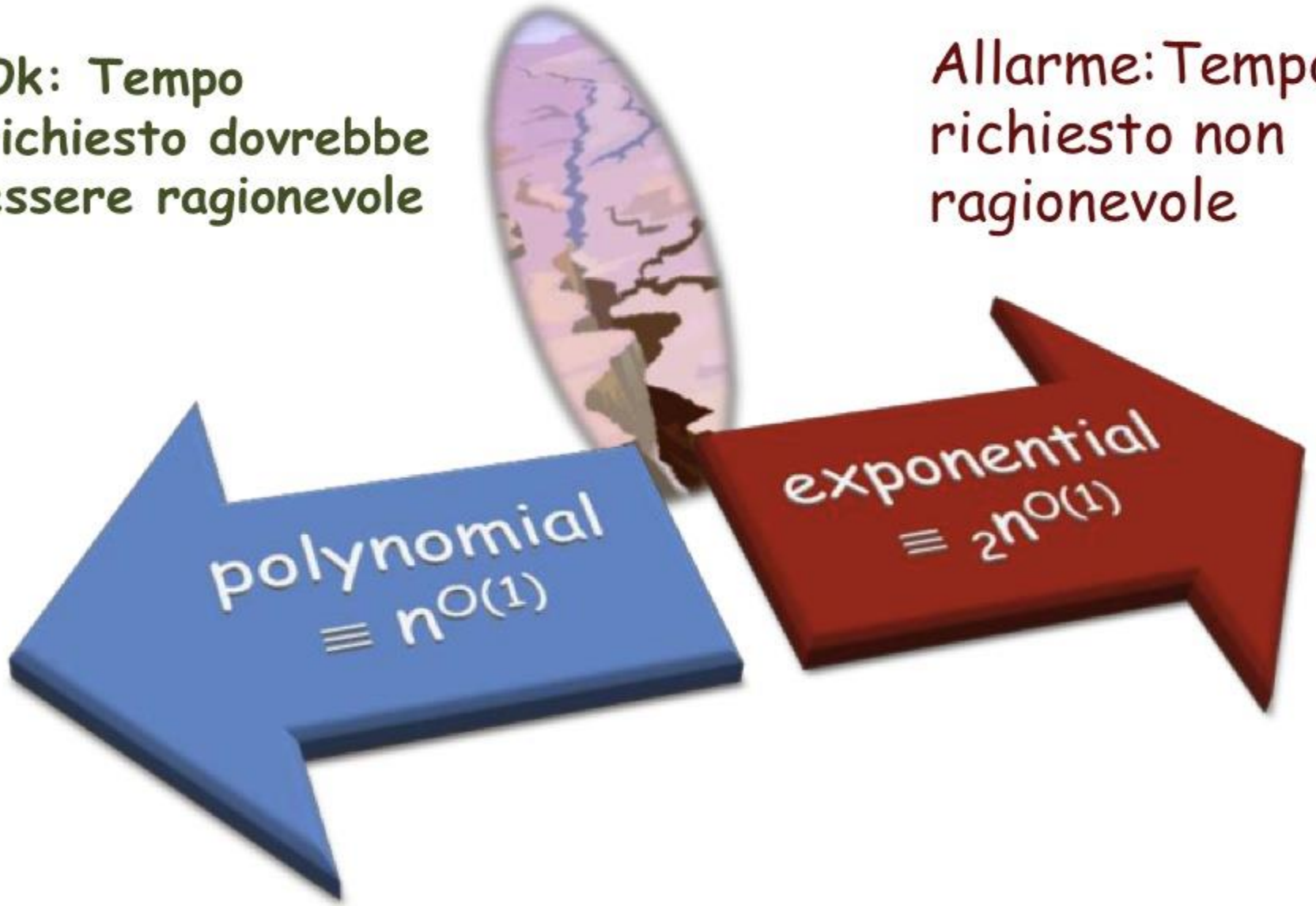
Algoritmi inefficienti: Utilizzano un tempo **esponenziale** su **qualche** input

Nota.
Definizione indipendente da sviluppo tecnologico

Cosa rende un problema facile o meno?

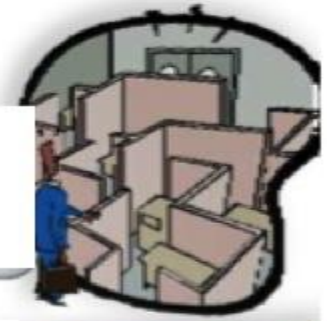
Ok: Tempo richiesto dovrebbe essere ragionevole

Allarme: Tempo richiesto non ragionevole



Il problema è trattabile o meno?

SI, ecco un algoritmo che lo risolve!



No, ed è anche possibile dimostrarlo

Nessuna delle due.

Esempi del terzo tipo

- **Problema del commesso viaggiatore**
- **Programmazione lineare intera**: esiste una soluzione intera di un sistema di equazioni del tipo

$$
\begin{aligned}
-x + y &\leq 1 \\
3x + 2y &\leq 12 \\
2x + 3y &\leq 12 \\
x, y &\geq 0 \\
x, y &\in \mathbb{Z}
\end{aligned}
$$

Il problema è trattabile o meno?

SI, ecco un algoritmo che lo risolve!

No, ed è anche possibile dimostrarlo

Nessuna delle due.

Cosa rende un problema "facile" o meno?

Le classi P e NP

Definizione (informale) della classe P:

insieme di problemi risolubili in tempo polinomiale da una macchina di Turing deterministica

Tesi Church-Turing ==>

insieme di problemi che ammettono un' algoritmo efficiente

Le classi P e NP

Definizione (*informale*) della classe NP:

insieme di problemi per cui non si conosce un algoritmo efficiente, ma che

ammettono un' algoritmo efficiente di *verifica di una soluzione fornita*

Problemi NP-completi

- *Travelling Salesman Problem,*
- *3-coloring di grafi,*
- *Scheduling Multiprocessore,*
- *Folding Proteine,*
- *Programmazione lineare intera:*

  esiste soluzione intera per un sistema del tipo

$$\begin{aligned} x_1 - 4x_2 + x_3 &= 0 \\ x_1 + x_2 + x_3 &\leq 0 \\ x_1 + 7x_3 &\geq 0 \end{aligned}$$

Tutti risolvibili efficientemente o nessuno!

Argomenti di massima:

- Macchine a stati finiti
- Macchine di Turing
- Le classi P e NP

Risultati attesi

- **Computabilità**

  Saprete

  - che è impossibile dimostrare che un programma in C termina e nessun calcolatore futuro può cambiare la situazione

  - Come riconoscere un problema intrattabile se vi capita

Risultati attesi: Complessità

Saprete

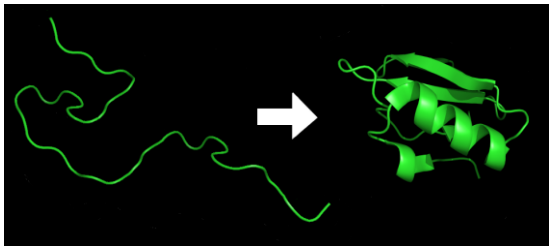
- Come riconoscere un problema computazionalmente difficile se vi capita

- Come modellare la computazione, in vari settori: Cellule/DNA, nel cervello, sistemi economici, social networks, ...

Es. Il folding delle proteine all'interno della cellula

Le proteine, sintetizzate come catene polipeptidiche che si estendono in modo spazialmente non strutturato, devono raggiungere una conformazione tridimensionale stabile per poter svolgere le loro funzioni biologiche.

Come determinarla? Come fanno le cellule?



Risultati attesi:

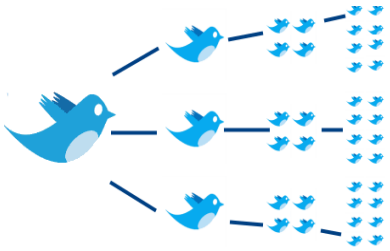
Saprete

- Come riconoscere un problema computazionalmente se vi capita

- Come modellare la computazione, in vari settori: Cellule/DNA, nel cervello, sistemi economici, social networks, ...

Es. *Perhaps nothing is as effective and efficient in spreading your message as a viral marketing campaign. The idea behind viral marketing is to inspire people to spread your message for you. It's been estimated that a successful viral campaign can have 500-1000 times more impact than a non-viral campaign.*

Come minimizzare i costi iniziali?



Informazioni Pratiche

ORARIO:

- **Martedì: 11:00 – 13:00**
- **Giovedì: 11:00 – 13:00**
- **Venerdì: 14:00 – 16:00**

N.B.: Tutte le lezioni sono ugualmente importanti!

Informazioni Pratiche

SITO WEB:
http://www.dia.unisa.it/professori/lg/ETC.html

di riferimento per il **materiale** relativo al corso

- copie delle **slides**, **esercizi**,
- date delle prove,
- comunicazioni varie,
- etc.

Suggerimenti

(per superare facilmente l'esame)

- Seguire il corso

  È più difficile imparare da soli dal libro di testo

  (ancora di più dalle slide!)

- Studiare lezione per lezione

  I concetti del corso richiedono un pò di tempo

  per essere assorbiti, non rimanete indietro!

Suggerimenti

(per superare facilmente l'esame)

- Studiare dal **libro di testo**

  Non come se fosse un romanzo giallo

  L'obiettivo non è avere risposte, ma imparare i concetti e le tecniche

- Fare gli esercizi

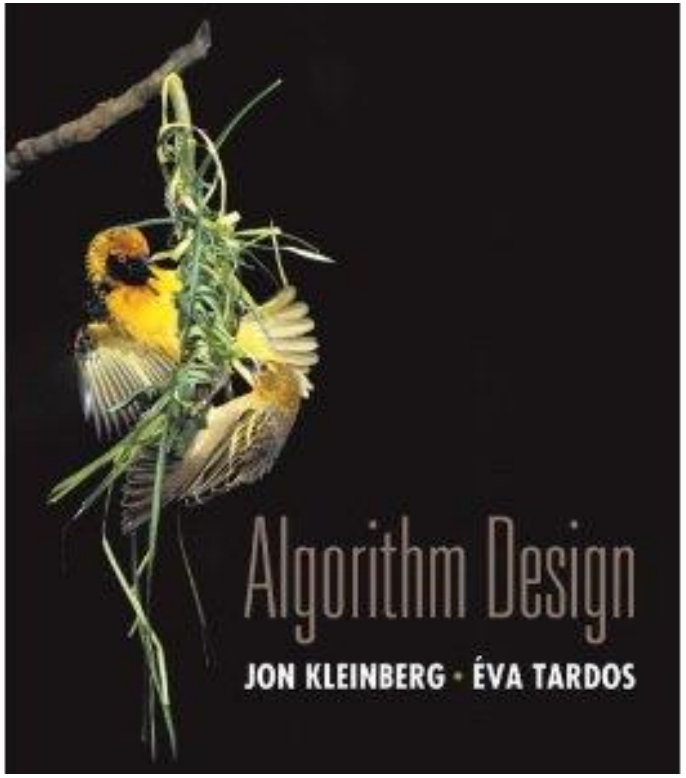
  provate sempre a pensare come risolvere un problema prima di sentire la risposta

Testo

Michael Sipser, **INTRODUZIONE ALLA TEORIA DELLA COMPUTAZIONE**, MAGGIOLI, 2016

Testo

- Jon Kleinberg, Eva Tardos, **Algorithm Design,** Pearson (solo Capitolo 8)



Prove di Esame

- Prova scritta con esercizi e teoria

  (nessun materiale ammesso)

- Eventuale prova orale

- Requisito minimo: 48% del totale

Prove in Itinere

- 2 prove
- stesse modalità delle prove d'esame

Progamma sintetico

- **Nozioni preliminari**
- Automi Finiti
- Macchine di Turing
- Limiti delle macchine di Turing
- La tesi di Church-Turing
- Le classi P e NP