



SAPIENZA  
UNIVERSITÀ DI ROMA

UNIVERSITÀ "SAPIENZA" DI ROMA  
FACOLTÀ DI INFORMATICA

---

# Introduzione agli Algoritmi

---

*Author*  
Simone Bianco

23 febbraio 2022

# Indice

<b>0</b>	<b>Introduzione</b>	<b>1</b>
<b>1</b>	<b>Algoritmi, Efficienza e RAM</b>	<b>2</b>
1.1	Algoritmi e Strutture Dati . . . . .	2
1.2	Efficienza di un algoritmo . . . . .	3
1.2.1	Random Access Machine (RAM) . . . . .	4
1.2.2	Misura di Costo Uniforme . . . . .	5

# Capitolo 0

## Introduzione

Si può facilmente osservare che al giorno d'oggi l'informatica **permea la nostra vita quotidiana**, sia quando essa è direttamente percepibile, sia quando è invisibile.

L'informatica viene spesso erroneamente considerata una mera attività pratica, per svolgere la quale è sufficiente un approccio diletteristico e per cui non è necessaria una vera professionalità. Nulla di più inesatto: in realtà l'informatica è una **disciplina scientifica**.

Essa non può essere considerata una sorta di “scienza dei calcolatori”, poiché **i calcolatori** (o elaboratori) ne **sono solo uno strumento**: l'informatico può anche lavorare solamente con carta e penna. In realtà, l'informatica, intesa come disciplina scientifica, non coincide con alcuna delle sue applicazioni.

In questo corso verranno date le definizioni di **algoritmo**, **strutture dati**, **efficienza**, **problemi computazionali** ed **ottimizzazione** di essi. Viene inoltre fornito un **modello teorico di calcolatore** che consentirà di paragonare tra loro algoritmi diversi che risolvono lo stesso problema.

# Capitolo 1

## Algoritmi, Efficienza e RAM

### 1.1 Algoritmi e Strutture Dati

La definizione di informatica proposta dall'ACM (**Association for Computing Machinery**), nonché una delle principali organizzazioni scientifiche di informatici di tutto il mondo, è la seguente: *"L'informatica è la scienza degli algoritmi che descrivono e trasformano l'informazione: la loro teoria, analisi, progetto, efficienza, realizzazione e applicazione."*

Gli **algoritmi**, dunque, sono un concetto fondamentale per l'informatica, fino ad esserne il fulcro. Ma cosa intendiamo per algoritmo?

#### Definition 1. Algoritmo

Un **algoritmo** è "una sequenza di comandi elementari ed univoci che terminano in un **tempo finito** ed operano su **strutture dati**".

Un comando viene definito **elementare** quando **non può essere scomposto** in comandi più semplici. I comandi elementari sono quindi **univoci** e possono essere interpretati in un solo modo.

Se un algoritmo è **ben specificato**, chi (o ciò che) lo esegue non ha bisogno di pensare, deve solo eseguire con precisione i passi elencati nell'algoritmo nella sequenza in cui appaiono. Se un calcolatore esegue un algoritmo e l'output è errato, **la colpa non è del calcolatore, ma del progettista**.

Prima di poter risolvere un problema abbiamo, ovviamente, bisogno di pensare ad un modo per poter **gestire i dati** che vengono utilizzati dall'algoritmo stesso. A tal fine, sarà necessario definire le opportune **strutture dati** su cui opererà l'algoritmo, ossia gli strumenti necessari per **organizzare** e **memorizzare** i dati veri e propri, semplificandone l'accesso e la modifica.

È importante sottolineare che **non esiste una struttura dati che sia adeguata per ogni problema**, dunque è necessario conoscere proprietà, vantaggi e svantaggi delle principali strutture dati in modo da poter scegliere di volta in volta quale sia quella **più adatta al problema**.

La scelta della struttura dati da adottare nella soluzione di un problema è un **aspetto fondamentale** per la risoluzione del problema stesso, al pari del progetto dell'algoritmo stesso. Per questa ragione, gli algoritmi e le strutture dati fondamentali vengono sempre studiati e illustrati assieme.

## 1.2 Efficienza di un algoritmo

Un aspetto fondamentale che va affrontato nello studio degli algoritmi è la loro **efficienza**, cioè la quantificazione delle loro **esigenze in termini di tempo e di spazio**, ossia tempo di esecuzione e quantità di memoria richiesta.

La scelta di un algoritmo rispetto ad altro, nel caso i due algoritmi portino allo stesso risultato, è molto importante:

- I calcolatori sono molto veloci, ma non infinitamente veloci
- La memoria è economica e abbondante, ma non è né gratuita né illimitata.

Un parametro fondamentale per la scelta dell'algoritmo è proprio la quantità di risorse spazio-tempo utilizzate. Nelle sezioni successive vedremo il concetto di **costo computazionale** degli algoritmi in termini di numero di operazioni elementari e quantità di spazio di memoria necessario in funzione della dimensione dell'input.

### Esempio di valutazione dell'efficienza

Immaginiamo di voler risolvere il seguente problema: vogliamo **ordinare una lista di  $n = 10^6$  numeri interi**. Vista l'enorme quantità di numeri, decidiamo di far svolgere questo compito ad un elaboratore. A nostra disposizione abbiamo **due calcolatori**:

- Un **calcolatore veloce**, che chiameremo **V**, in grado di svolgere  $10^9$  operazioni/sec
- Un **calcolatore lento**, che chiameremo **L**, in grado di svolgere  $10^7$  operazioni/sec

Immaginiamo di essere in grado di saper sviluppare solo **due algoritmi di ordinamento** (di cui per ora non vedremo il funzionamento, ma solo le specifiche temporali):

- L'algoritmo **Insertion Sort**, richiedente  $2n^2$  operazioni (**più lento**)
- L'algoritmo **Merge Sort**, richiedente  $50n \cdot \log(n)$  operazioni (**più veloce**)

Ci chiediamo se la maggior velocità del calcolatore V sia in grado di **contro-bilanciare** la maggior lentezza dell'algoritmo IS. Proviamo quindi a calcolare il costo temporale di entrambe le scelte (**ATTENZIONE**: con  $\log$  intendiamo il **logaritmo in base 2**):

$$V(IS) = \frac{2 \cdot (10^6)^2 \text{ operazioni}}{10^9 \text{ operazioni/sec}} = 2000 \text{ sec} \approx 33 \text{ min}$$

$$L(MS) = \frac{50 \cdot 10^6 \cdot \log(10^6) \text{ operazioni}}{10^7 \text{ operazioni/sec}} \approx 100 \text{ sec} \approx 1.5 \text{ min}$$

Notiamo quindi che, nonostante la differenza di caratteristiche hardware, **la scelta dell'algoritmo è cruciale per l'efficienza**.

Per ricalcare maggiormente il concetto, proviamo ad aumentare l'input a  $n = 10^7$

$$V(IS) = \frac{2 \cdot (10^7)^2 \text{ operazioni}}{10^9 \text{ operazioni/sec}} \approx 55.5 \text{ ore} \approx 2.3 \text{ giorni}$$

$$L(MS) = \frac{50 \cdot 10^7 \cdot \log(10^7) \text{ operazioni}}{10^7 \text{ operazioni/sec}} \approx 19.5 \text{ min}$$

Aumentando l'input di un solo ordine di grandezza, la **differenza** in termini di costi temporali dei due algoritmi è **abissale**.

### 1.2.1 Random Access Machine (RAM)

Nell'esempio precedentemente visto, abbiamo considerato **due macchine diverse**, dove una era più performante dell'altra. Ciò non è un fattore da ignorare, poiché ovviamente le caratteristiche hardware dell'elaboratore **influiscono** direttamente sulle **performance dell'algoritmo**: se nell'esempio precedente calcolassimo  $V(MS)$  con  $n = 10^6$ , il tempo impiegato dall'algoritmo sarebbe circa **1 secondo**, rispetto ai **100 secondi** impiegati da  $L(MS)$ .

Per poter valutare la **vera efficienza** di un algoritmo, dunque, è necessario quantificare le risorse che esso richiede per la sua esecuzione senza che tale analisi sia **influenzata da una specifica tecnologia** che, inevitabilmente, col tempo **diviene obsoleta**. Dunque, è necessario valutare l'algoritmo come se venisse eseguito da una **macchina astratta** rispettante queste caratteristiche, ossia la **Random Access Machine** (anche chiamata **Modello RAM**).

#### Definition 2. Random Access Machine

La **Random Access Machine (RAM)** è una macchina astratta, la cui validità e potenza concettuale risiede nel fatto che **non diventa obsoleta** con il progredire della tecnologia.

Le caratteristiche del modello RAM sono:

- Un **singolo processore** che esegue le operazioni **sequenzialmente**
- Esistono **solo operazioni elementari** e l'esecuzione di ciascuna delle quali richiede per definizione un **tempo costante** (es.: operazioni aritmetiche, letture, scritture, salto condizionato, ecc.)
- Esiste un **limite alla dimensione** di ogni valore memorizzato ed al numero complessivo di valori utilizzati, dipendente dalle dimensioni delle word in memoria

### 1.2.2 Misura di Costo Uniforme

Sia  $d$  la **dimensione di bit di ogni parola** contenuta in memoria. Se è soddisfatta l'ipotesi che ogni dato in input sia un valore **minore** di  $2^d$ , ciascuna operazione elementare sui dati del problema verrà eseguita in un **tempo costante**. In tal caso si parla di **misura di costo uniforme**.

Tale criterio **non è sempre realistico**: se un dato del problema non rispetta tale ipotesi, esso dovrà essere comunque memorizzato. In tal caso, sarà necessario usare **più parole di memoria** e, di conseguenza, anche le operazioni elementari su di esso dovranno essere reiterate per tutte le parole di memoria che lo contengono, richiedendo quindi un tempo non più costante. Per questo motivo, in ambito scientifico viene utilizzata la **misura di costo logaritmica**, più realistica rispetto a quella uniforme. Tuttavia, in questo corso essa **non verrà analizzata**.

#### Esempio di costo uniforme

Analizziamo il seguente codice:

```
def PotenzaDi2(n)
    x = 1
    for i in range(n):
        x = x*2
    return x
```

Il tempo di esecuzione totale è **proporzionale ad  $n$** , poiché si tratta di un **ciclo eseguito  $n$  volte**, dove ad ogni iterazione vengono compiute tre operazioni, ciascuna di **costo unitario**:

- Viene incrementato il contatore relativo al ciclo for
- Viene calcolato  $x \cdot 2$
- Viene assegnato il risultato del calcolo ad  $x$