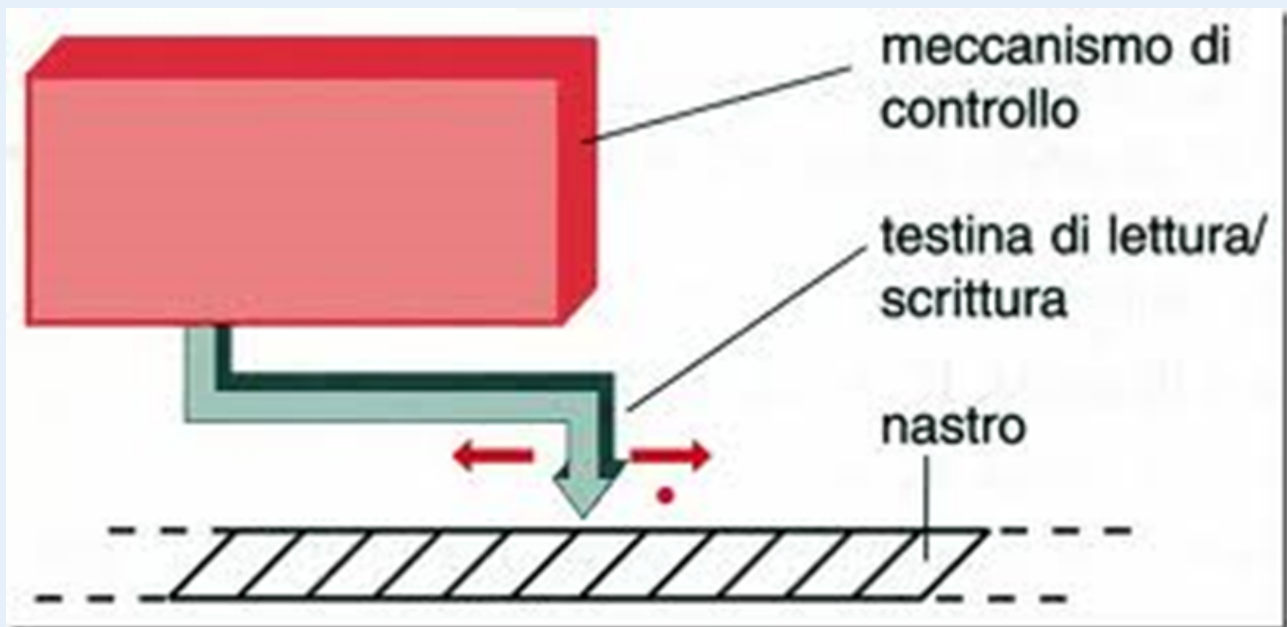


Modelli di Calcolo e Notazione Asintotica

Alla fine della lezione precedente di siamo chiesti se fosse corretto calcolare la complessità di un algoritmo contando il numero di linee di codice eseguite. La risposta è che dipende.

Dipende infatti dal modello di calcolo che decidiamo di usare.

Esempio storico di modello di calcolo: La macchina di Turing



Non è il modello di calcolo che useremo per due motivi:

1. **Basso livello:** Somiglia troppo poco ai calcolatori reali su cui girano i programmi.
2. Utile per parlare di **calcolabilità**, ma meno per parlare di **efficienza**.

Per questo utilizzeremo una **RAM (Random Access Machine)** composta da:

- Un programma finito.
- Un nastro di ingresso e uno di uscita.
- Una memoria strutturata come un array.
- Una CPU che esegue istruzioni.

Generalmente è un'astrazione dell'architettura di Von Neumann.

Adesso possiamo basare l'analisi della complessità sul concetto di **passo elementare**.

I passi elementare sulla RAM sono:

- Istruzione ingresso/uscita.
- Operazioni aritmetico/logiche.
- Accesso/modifica del contenuto della memoria.

Ogni operazione ha ovviamente un costo, in base al criterio che usiamo essa avrà un valore diverso, ci sono 2 criteri:

- **Criterio di Costo Uniforme:** Tutte le operazioni hanno lo stesso costo, quindi la complessità temporale è misurata come numero di passi elementari eseguiti.
- **Criterio di Costo Logaritmico:** Il costo dell'operazione dipende dalla dimensione degli operandi dell'istruzioni, quindi un operando di valore x ha costo $\log(x)$, è un tipo di criterio che modella meglio la complessità di algoritmi numerici.

Quindi misuriamo il tempo di esecuzione di un algoritmo in funzione della dimensione n delle istanze, ovviamente istanze diverse della stessa dimensione potrebbero richiedere tempi diversi.

Possiamo distinguere la nostra analisi in **Caso Peggior**e e **Caso Medio**.

Caso Peggior

Sia $tempo(I)$ il tempo d'esecuzione di un algoritmo sull'istanza I , allora:

$$T_{worst}(n) = \max_{istanza\ di\ I\ di\ dimensione\ n} (tempo(I))$$

$T_{worst}(n)$ è il tempo d'esecuzione sulle istanze di ingresso che comportano più lavoro sull'algoritmo.

La particolarità è che il caso peggiore ci dà una garanzia sul tempo di esecuzione di ogni istanza.

Caso Medio

Sia $P(I)$ la probabilità di occorrenza dell'istanza I , allora:


$$T_{avg} = \sum_{istanza\ di\ I\ di\ dimensione\ n} (P(I)tempo(I))$$

T_{avg} è il tempo di esecuzione sulle istanze di ingresso "tipiche" per il problema.

Il problema principale è conoscere la distribuzione di probabilità sulle istanze, infatti quasi sempre si fanno delle assunzioni sulla probabilità.

Notazione Asintotica

Possiamo provare ad esprimere la complessità computazionale di un algoritmo esprimendola con una funzione $T(n)$.

 **L'idea è quella di descrivere la funzione in modo qualitativo, in modo tale da perdere un po' di precisione per guadagnare in semplicità, quindi:**

$T(n)$ = numero di passi elementari eseguiti su RAM nel caso peggiore su un'istanza di dimensione n

 **Esempio:**

$$T(n) = 71n^2 + 100\lfloor n/4 \rfloor + 7 \text{ se } n \text{ è pari}$$

$$T(n) = 70n^2 + 100\lfloor (n+1)/4 \rfloor + 5 \text{ se } n \text{ è dispari}$$

$$\text{Scriveremo } T(n) = \theta(n^2)$$

In maniera intuitiva $T(n)$ è proporzionale a n^2

Quindi ignoro:

1. Costanti moltiplicative
2. Termini di ordine inferiore


L'assunzione è che vogliamo guardare come si comporta l'algoritmo su istanze grandi.

Perché sull'istanze piccole ogni algoritmo è "buono".

Di seguito le principali notazioni asintotiche:

$O \quad \Omega \quad \Theta \quad o \quad \omega$

Notazione Asintotica O-grande

 **Definizione:** $f(n) = O(g(n))$ se \exists due costanti $c > 0$ e $n_0 > 0$ tali che:
 $0 \leq f(n) \leq c * g(n) \quad \forall n > n_0.$

Esempi:

Sia $f(n) = 2n^2 + 2n$, allora:

- $f(n) = O(n^3)$ per $c = 1, n_0 = 3$
- $f(n) = O(n^2)$ per $c = 3, n_0 = 3$
- $f(n) \neq O(n)$

$O(g(n)) = \{f(n): \exists \text{ due costanti } c > 0 \text{ e } n_0 > 0 \text{ tali che: } 0 \leq f(n) \leq c * g(n) \quad \forall n > n_0\}$

Di conseguenza la scrittura: $2n^2 + 4 = O(n^3)$

E' un abuso di notazione per: $2n^2 + 4 \in O(n^3)$

E questo vale per tutte le altre notazioni.

Inoltre è importante notare che:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \implies f(n) = O(g(n))$$

Il contrario vale se e solo se il limite del rapporto esiste ed è minore di infinito.

Notazione Asintotica Ω

 **Definizione:** $f(n) = \Omega(g(n))$ se \exists due costanti $c > 0$ e $n_0 > 0$ tali che:
 $f(n) \geq c * g(n) \geq 0 \quad \forall n > n_0.$

Esempio: $f(n) = 2n^2 - 3n$, allora $f(n) = \Omega(n)$ per $(c = 1, n_0 = 2)$


Come prima Ω è in realtà l'insieme delle funzioni f per cui vale la condizione precedente.

Inoltre è importante notare che:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty \implies f(n) = O(g(n))$$

Il contrario vale se e solo se il limite del rapporto esiste ed è maggiore di zero.

Notazione asintotica Θ

 **Definizione:** $f(n) = \Theta(g(n))$ se \exists due costanti $c_1, c_2 > 0$ e $n_0 > 0$ tali che:
 $c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n > n_0.$


Esempio: $f(n) = 2n^2 - 3n$, allora $f(n) = \theta(n^2)$ per $(c_1 = 1, c_2 = 2, n_0 = 3)$

Come prima Θ è in realtà l'insieme delle funzioni f per cui vale la condizione precedente.

Inoltre è importante notare che:

$$f(n) = \Theta(g(n)) \iff f(n) = \Omega(g(n)) \text{ e } f(n) = O(g(n))$$

Notazione asintotica o-piccolo

 **Definizione:** Data una funzione $g(n) : N \rightarrow R$, si denota con $o(g(n))$ l'insieme delle funzioni $f(n) : N \rightarrow R$:


$$o(g(n)) = \{f(n) : \forall c > 0, \exists n_0 : \forall n \geq n_0 \quad 0 \leq f(n) < c * g(n)\}$$

Praticamente l' o -piccolo è un sottoinsieme stretto dell' O -grande.

Di conseguenza:

$$f(n) = o(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Notazione asintotica ω

 **Definizione:** Data una funzione $g(n) : N \rightarrow R$, si denota con $\omega(g(n))$ l'insieme delle funzioni $f(n)$:

$$\omega(g(n)) = \{f(n) : \forall c > 0, \exists n_0 : \forall n \geq n_0 \quad 0 \leq c * g(n) < f(n)\}$$

Praticamente ω è un sottoinsieme di Ω .

Di conseguenza:

$$f(n) = \omega(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$$

Analogie e Proprietà delle notazione asintotiche

Possiamo vedere le notazioni asintotiche come delle analogie:

- O – grande sarebbe \leq ;
- Ω sarebbe \geq ;
- Θ sarebbe $=$;
- o – piccolo sarebbe $<$;
- ω sarebbe $>$;

Alcune proprietà:

- **Transitività:** $f(n) = O(g(n))$ e $g(n) = O(h(n)) \implies f(n) = O(h(n))$

Questa proprietà vale per tutte le notazioni.

- **Riflessività:** $f(n) = \Theta(f(n))$, $f(n) = O(f(n))$, $f(n) = \Omega(f(n))$
- **Simmetria:** $f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n))$
- **Simmetria trasposta:** $f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$,
 $f(n) = o(g(n)) \iff g(n) = \omega(f(n))$

Un'ulteriore proprietà utile per capire la velocità di una funzione:

$$\text{Se } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c > 0 \quad \text{allora} \quad f(n) = \Theta(g(n))$$

Le funzioni che ci interessano quando utilizziamo le notazioni asintotiche sono funzioni:

- **Polinomiali**
- **Esponenziali**
- **Logaritmiche**
- **Fattoriali**

Velocità asintotica di funzioni composte

Date $f(n)$ e $g(n)$ la velocità ad andare a infinito della funzione $f(n) + g(n)$ è la velocità della più veloce fra $f(n)$ e $g(n)$

Esempi:

$$1) n^3 + n = \Theta(n^3)$$

$$2) n + \log^{10}(n) = \Theta(n)$$

Date $f(n)$ e $g(n)$:

- La velocità ad andare a infinito della funzione $f(n)g(n)$ e la velocità di $f(n)$ "più" la velocità di $g(n)$.
- La velocità ad andare a infinito della funzione $f(n)/g(n)$ e la velocità di $f(n)$ "meno" la velocità di $g(n)$.

Esempio:

$$\frac{n^3 \log(n) + \sqrt{n} * \log^3(n)}{n^2 + 1} = \Theta(n * \log(n))$$

Un'ultima cosa da dire quando si fa la complessità di alcuni algoritmi conviene calcolare un **Upper bound** e un **Lower bound**, per esempio in fibonacci3.

Perché la notazione asintotica è così utile:

- **Misura indipendente** dall'implementazione dell'algoritmo e **dalla macchina reale** su cui è eseguito.
- I "dettagli" nascosti sono **poco rilevanti** quando n è grande.
- Fare un'analisi dettagliata dei passi realmente eseguiti è molto difficile e **non direbbe molto di più rispetto alla notazione asintotica**.
- In pratica descrivono molto bene la velocità degli algoritmi.