In algoritmica (o scienza degli algoritmi) la **complessità** è una funzione che indica il numero di **operazioni elementari** impiegate per risolvere un problema in funzione della dimensione n dei **dati** del problema.

n è il numero di caratteri (tipicamente bit o cifre decimali) con cui si rappresentano i dati di ingresso e uscita. Ogni operazione elementare richiede tempo costante, quindi indipendente da n.

In particolare:

- la complessità CP(n) di un problema P è il minimo numero di operazioni elementari necessarie a risolverlo
- la complessità CA(n) di un algoritmo A per la soluzione di P è il numero di operazioni elementari che esso esegue

CP(n) ≤ CA(n) costituiscono rispettivamente un limite inferiore e un limite superiore di complessità, in quanto:

- il problema P non si può risolvere con meno operazioni elementari di CP(n)
- trovato un algoritmo A, non interessano gli algoritmi che richiedono più operazioni elementari di CA(n).

Se CP(n) = CA(n), A è un algoritmo ottimo per P.

Le funzione di complessità CP e CA si valutano in ordine di grandezza, a parte casi particolari in cui come vedremo si può stabilirne un valore esatto.

Vedremo sotto i due ordini di grandezza più significativi in informatica. Introduciamoli informalmente con due esempi semplicissimi.

ADDIZIONE TRA DUE NUMERI DI m CIFRE

m = 6

```
 1  6  5  2  8  7  +
 5  4  3  9  4  2  =
 ----------------
 7  0  9  2  2  9
```

- La dimensione dei dati n è uguale a 2m. Questo è un valore per il limite inferiore CP(n) perché tutte le cifre degli addendi devono essere esaminate
- Il numero delle operazioni elementari eseguite dall'algoritmo (addizioni di due cifre con un eventuale resto precedente, e registrazione del risultato) è proporzionale a n. Questo è un valore per il limite superiore CA(n).

Nel problema dell'addizione e del suo algoritmo elementare CP(n) e CA(n) sono entrambi proporzionali a n, quindi CP(n) = CA(n) in ordine di grandezza.
L'algoritmo elementare di addizione è ottimo e non è necessario cercarne altri

ma vediamo un secondo esempio forse inaspettato.

MOLTIPLICAZIONE TRA DUE NUMERI DI m CIFRE

m = 3

```
          2  8  7 ✖
          5  4  3 =
  ---------------
          8  6  1
    1  1  4  8
 1  4  5  5
  ---------------
 1  5  7  8  4  1
```

La dimensione dei dati n è uguale a 2m. Anche ora questo è un valore per il limite inferiore CP(n) perché tutte le cifre dei fattori devono essere esaminate

Le operazioni elementari eseguite dall'algoritmo sono:

- n^2 moltiplicazioni tra due cifre ed eventuale addizione al resto precedente, per la costruzione dei prodotti parziali (righe della tabella)

- n^2 addizioni nelle colonne dei prodotti parziali

- registrazione delle $2n$ cifre del risultato

Il totale $2n^2 + 2n$ è il valore del limite superiore CA(n), di ordine di grandezza maggiore di quello n del limite inferiore CP(n).

Diverse situazioni sono possibili:

1. esiste un limite inferiore CP(n) più alto dell'ordine n trovato

2. esiste un limite superiore CA(n) più basso dell'ordine n^2 trovato (cioè un algoritmo più efficiente per la moltiplicazione)

3. Sono veri entrambi i casi

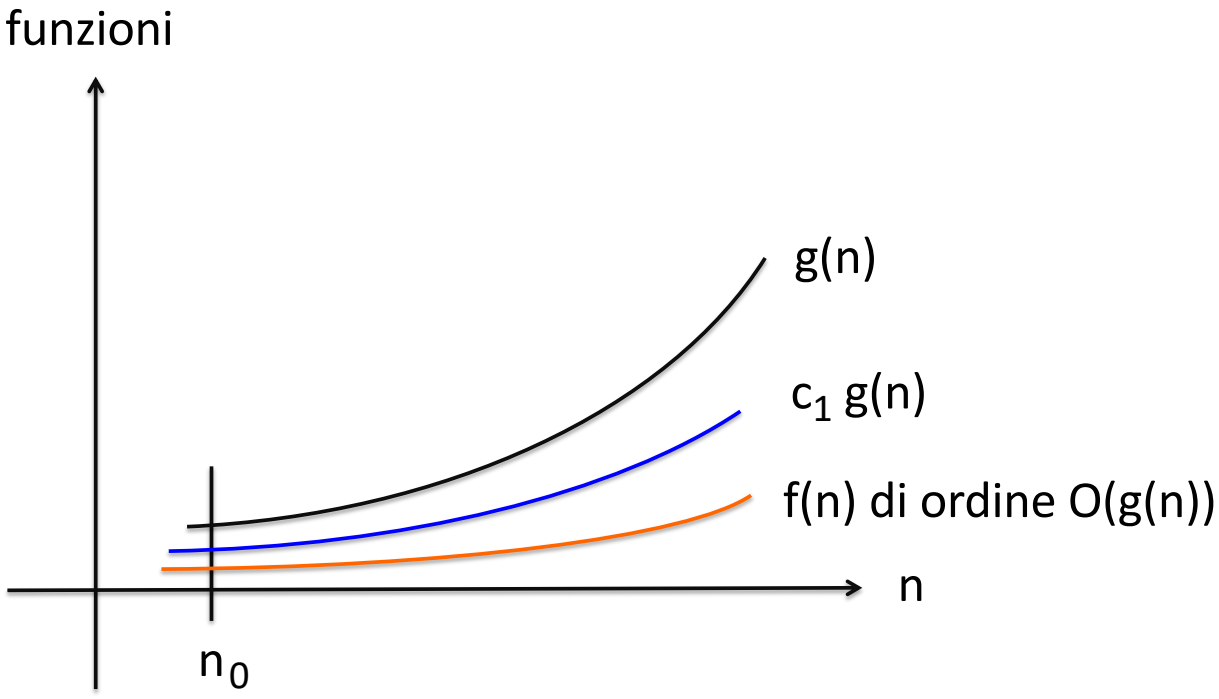
Attualmente la situazione è la seguente:

- non è stato trovato alcun limite inferiore CP(n) di ordine più alto di n
- esistono algoritmi di moltiplicazione più efficienti di quello elementare, cioè con CA(n) inferiore a n^2, ma nessuno di essi raggiunge n

Dal punto di vista computazionale il problema della moltiplicazione è ANCORA APERTO

Notazione O

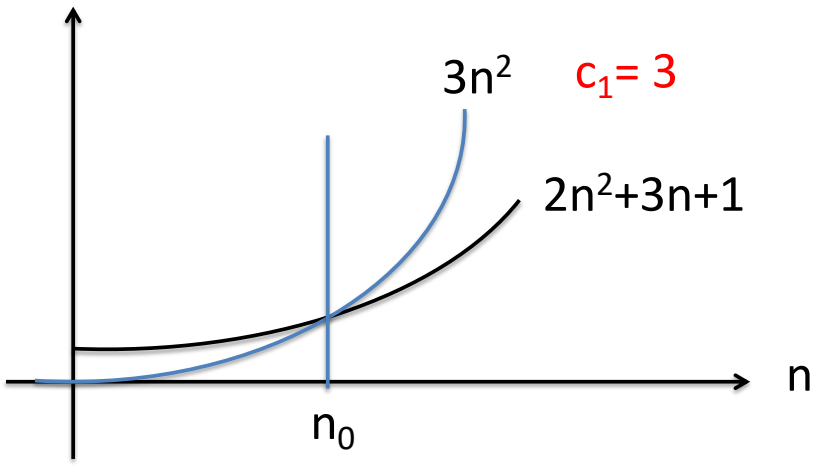
f(n) **è di ordine O**(g(n)) se esistono due costanti positive c_1, n_0, tali che $0 \leq f(n) \leq c_1 g(n)$ per ogni $n \geq n_0$.



Cioè, al crescere di n e a partire da un valore n_0 , la funzione $f(n)$ non sale al di sopra di $g(n)$ a meno di una costante moltiplicativa c_1:

dunque nella funzione $g(n)$ non si indicano le costanti moltiplicative

Per esempio $f(n) = 2n^2+3n+1$ è di ordine $\mathbf{O}(n^2)$
(contano solo i termini di grado massimo)



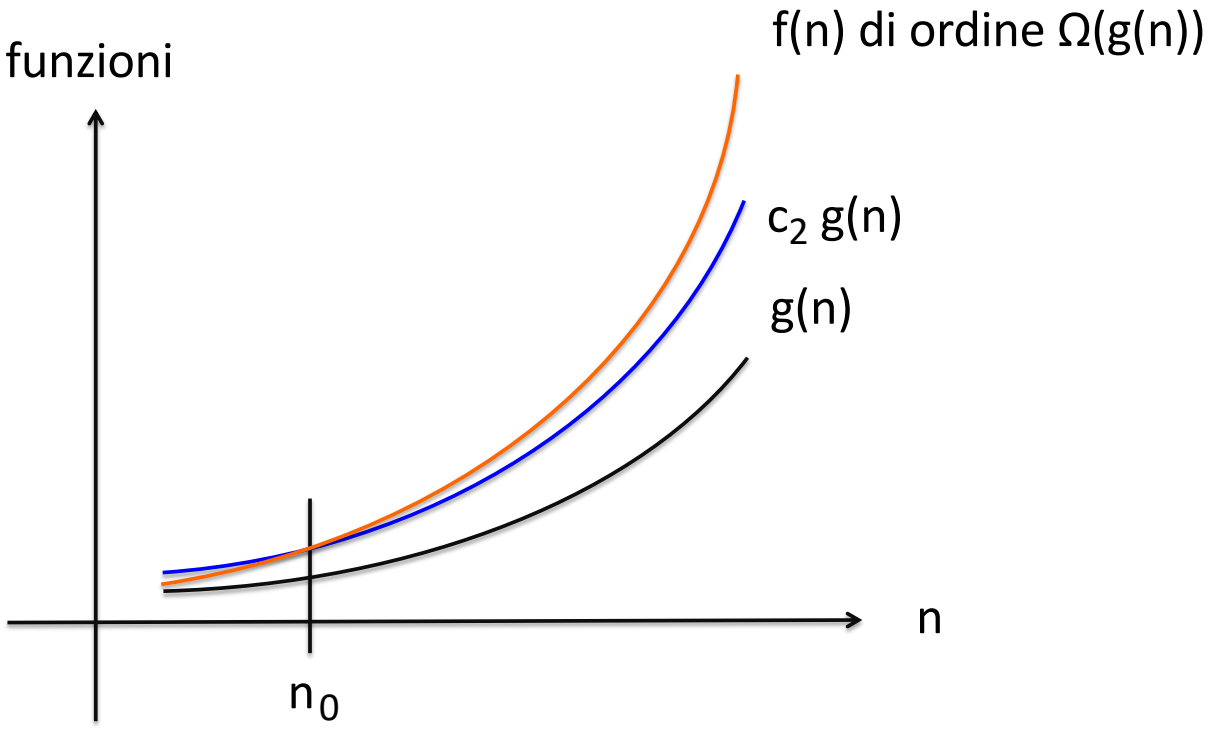
La notazione O(g(n)) si impiega per indicare il tempo di un algoritmo:

- di cui **non** si conosce compiutamente il comportamento, ma che si sa che non può superare g(n);

- oppure che **non** si comporta allo stesso modo per tutti gli insiemi di dati di dimensione n che gli si presentano, ma per alcuni richiede tempo proporzionale a g(n)), per altri meno.

Notazione Ω

f(n) **è di ordine Ω**(g(n)) se esistono due costanti positive c_2, n_0, tali che $0 \leq c_2 g(n) \leq f(n)$ per ogni $n \geq n_0$.



Cioè, al crescere di n e a partire da un valore n_0, la funzione $f(n)$ non scende al di sotto di $g(n)$ a meno di una costante moltiplicativa.

Anche qui contano solo i termini di grado massimo.

La notazione Ω si impiega per indicare il limite inferiore al tempo di soluzione di un problema.

Notare l'importante differenza nell'impiego dei due ordini O e Ω. Il primo è relativo al comportamento di un particolare algoritmo di soluzione, il secondo alla natura intrinseca del problema e si applica quindi a *tutti i suoi algoritmi* di soluzione