Complessità degli algoritmi

L'esecuzione di un programma implica un costo economico, dovuto all'utilizzo delle risorse (memoria, traffico sulla rete, spazio su disco, ecc.) e di tempo di elaborazione. In genere si parla di

- **Complessità spaziale** per intendere l'utilizzo delle risorse da parte di un programma.
- **Complessità temporale** per intendere il tempo di esecuzione di un programma.

Nello sviluppo di un algoritmo è particolarmente importante la complessità temporale.
In informatica per calcolare la complessità computazionale di un algoritmo si utilizza l'analisi asintotica. Tuttavia, l'analisi asintotica è uno strumento della matematica che si applica alle funzioni, mentre il tempo di esecuzione non lo è. Quindi, per usare l'analisi asintotica devo trasformare il tempo di esecuzione dell'algoritmo in una funzione T(n) in funzione della dimensione n dei dati input. In genere la funzione T(n) misura il numero di comandi eseguiti dall'algoritmo. Data un'istanza di dimensione n, nel caso peggiore l'algoritmo ha una complessità temporale O(f(n)) se T(n)=O(f(n)). Dove n è il numero delle righe eseguite (dimensione n dei dati) mentre f(n) è un limite superiore del tempo di esecuzione dell'algoritmo nell'ipotesi peggiore. Nell'analisi asintotica il simbolo O è detto "o grande". In ogni caso, per valutare la complessità di un algoritmo si utilizza sempre l'ipotesi del caso peggiore.

Limite superiore asintotico (O-grande):

$\exists\ c > 0, n_0 > 0\ tc\ f(n) \leq cg(n)\ \forall n \geq n_0$, $g(n)$ è detto limite superiore asintotico di $f(n)$ e scriviamo $f(n) = O(g(n))$.

Limite inferiore asintotico (Ω-grande):

$\exists\ c > 0, n_0 > 0\ tc\ f(n) \geq cg(n)\ \forall n \geq n_0$, $g(n)$ è detto limite inferiore asintotico di $f(n)$ e scriviamo $f(n) = \Omega(g(n))$.

Esempi:

1. Sia $g(n) = n^2$ e sia $f(n) = 3n^2 + 5$. Allora:
   $4g(n) = 4n^2 = 3n^2 + n^2 \geq 3n^2 + g(n)\ ; \forall n \geq 3\ \ 4g(n) > f(n) \Rightarrow f(n) = O(g(n))$

2. Sia $g(n) = n^2$ e sia $f(n) = \frac{n^2}{2} - 7$. Allora:

$$\frac{g(n)}{4} = \frac{n^2}{4} = \frac{n^2}{2} - \frac{n^2}{4} \leq \frac{n^2}{2} - g(n) \; ; \forall n \geq 6 \;\; \frac{g(n)}{4} < f(n) \Rightarrow f(n) = \Omega(g(n))$$

Il limite superiore asintotico (O-grande) denota il caso pessimo: l'algoritmo compie <u>al più</u> n passi ($f(n) \leq cg(n) \; \forall n \geq n_0$).

Il limite inferiore asintotico (Ω-grande) denota il caso ottimo: l'algoritmo compie <u>come minimo</u> n passi ($f(n) \geq cg(n) \; \forall n \geq n_0$).

Un <u>problema</u> ha complessità $O(f(n))$ se esiste almeno un algoritmo che ha complessità $O(f(n))$.

Un <u>problema</u> ha complessità $\Omega(f(n))$ se tutti i possibili algoritmi che lo risolvono hanno complessità $\Omega(f(n))$.

La scala della complessità

La complessità di un algoritmo può essere classificata in una delle seguenti categorie:

$O(n!)$: complessità fattoriale (complessità massima)
$O(k^n)$ con $k > 0$: complessità esponenziale
$O(n^k)$ con $k > 0$: complessità polinomiale
$O(n^3)$: complessità cubica
$O(n^2)$: complessità quadratica (complessità media)
$O(n \log n)$: complessità pseudolineare
$O(n)$: complessità lineare (complessità minima)
$O(\log n)$: complessità logaritmica
$O(k)$: complessità costante - es. $O(1)$