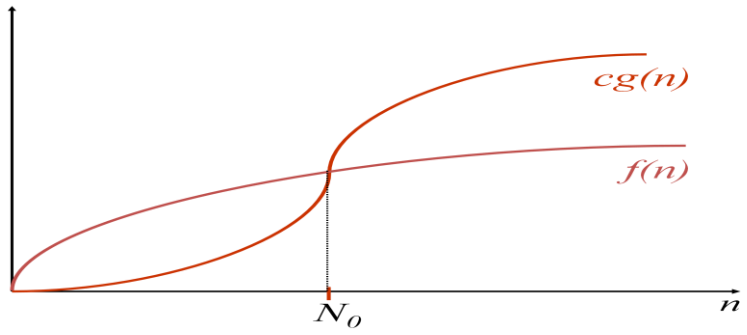
DOMANDA: Che cosa è la complessità asintotica di un algoritmo?

RISPOSTA:
La complessità di un algoritmo è una misura astratta della quantità di risorse computazionali necessarie per la sua esecuzione.
In particolare, si parla di complessità di spazio e di complessità di tempo, che si riferiscono alle due principali risorse computazionali: lo spazio di memoria e il tempo dell'esecutore.
Formalmente, la complessità di spazio e la complessità di tempo, indicate rispettivamente con S(n) e con T(n), sono funzioni della dimensione computazionale n del problema da risolvere.
La dimensione computazionale di un problema è una misura astratta del numero dei dati, o in generale di informazioni, che definiscono il problema, o meglio una particolare *istanza* del problema.
S(n) è il size di tutte le strutture dati (di input, locali e di output) che devono essere utilizzate dall'algoritmo durante la sua esecuzione per risolvere una istanza di dimensione n del problema.
T(n) è il numero di operazioni dominanti (di input, locali e di output) che devono essere eseguite dall'algoritmo per risolvere una istanza di dimensione n del problema.
Il termine complessità asintotica si riferisce al comportamento asintotico, cioè al tendere di n all'infinito, delle funzioni S(n) e T(n). Si noti che S(n) e T(n), escludendo il caso di algoritmi che implementano una formula chiusa e la cui complessità è quindi costante, sono funzioni che tendono a +infinito per n che tende a +infinito.
La notazione asintotica delle funzioni di complessità ha lo scopo di evidenziare in modo sintetico il loro comportamento asintotico, indicando solo i termini che crescono più velocemente rispetto agli altri eventuali termini che costituiscono l'espressione delle funzioni S(n) e T(n).
La notazione asintotica dell' "O" (che si legge "O grande") è formalmente definita nel seguente modo.
Siano f(n) e g(n) due funzioni definite (dominio) sui numeri naturali e a valori (codominio) sui numeri naturali; f(n) e g(n) sono funzioni non decrescenti, cioè sono costanti oppure sono crescenti:
allora f(n) = O(g(n)) significa che esistono due costanti (cioè non dipendenti da n) c e N_0, non negative, tali che f(n) <= c g(n) per n >= N_0.
In altre parole, con la notazione f(n) = O(g(n)) intendiamo affermare che la funzione f(n) tende all'infinito con la stessa rapidità o con rapidità inferiore rispetto a una scalatura opportuna della funzione g(n). Il termine scalatura di una funzione indica semplicemente il prodotto della funzione per un numero non negativo.
L'interpretazione grafica di tale definizione è immediata: il grafico della funzione f(n) non è al di sopra del grafico della funzione scalata c g(n) per tutti i valori di n a partire da N_0 in poi.



Vediamo alcuni esempi riferiti alla funzione complessità di tempo.
$T(n)=1453 \Rightarrow T(n)=O(1)$ cioè la complessità di tempo è costante
$T(n)=10n+1000 \Rightarrow T(n)=O(n)$ cioè la complessità di tempo è lineare
$T(n)=100000(n+1) \Rightarrow T(n)=O(n)$ cioè la complessità di tempo è lineare
$T(n)=30n^2+100n+233 \Rightarrow T(n)=O(n^2)$ cioè la complessità di tempo è quadratica

$$T(n)=\frac{n\cdot(n-1)}{2}=O(n^2)$$

In generale, se T(n) è un polinomio di grado k, allora $T(n)=O(n^k)$

$T(n)=\frac{1}{2}n+1000\log(n) \Rightarrow T(n)=O(n)$ cioè la complessità di tempo è lineare

$T(n)=1.5^n+n^{1000} \Rightarrow T(n)=O(1.5^n)$ cioè la complessità di tempo è esponenziale

$T(n)=1.5^n+n^{1000} \Rightarrow T(n)=O(1.5^n)=O(2^n)=O(10^n)$ cioè la complessità di tempo è esponenziale.