analisi della **complessità** degli **algoritmi**

(da: wikipedia Teoria_della_complessità_computazionale e Algoritmo_di_ordinamento)

L'**analisi di complessità** è un metodo formale per prevedere le risorse di calcolo richieste da un algoritmo (cioè il tempo di esecuzione e la quantità di memoria utilizzata).

- l tempo però non è un tempo fisico o misurato sperimentalmente su un processore, ma dipende dal numero di passi da eseguire: l'analisi di complessità è **indipendente dalla macchina**. Si assume che la macchina lavori in modo sequenziale (architettura di Von Neumann) e non parallelo.

- L'analisi è **indipendente da quali dati in ingresso** ci sono ma è **dipendente** solo **da quanti dati in ingresso** ci sono, cioè più in generale dalla **dimensione *n*** del problema.

Output prodotti dall'analisi di complessità:

- $S(n)$: memoria richiesta (spazio)
- $T(n)$: tempo di esecuzione

Per misurare l'efficienza di un algoritmo in maniera univoca, bisogna definire una metrica che sia indipendente dalle tecnologie utilizzate, altrimenti uno stesso algoritmo potrebbe avere efficienza diversa a seconda della tecnologia sulla quale viene eseguito.

Per questo motivo si usa fare riferimento ad un modello di calcolo generico: la **macchina di Turing**.
Ai fini della classificazione dei problemi, qualunque modello di calcolo scelto (si può parlare anche di computer reali) si comporta come la macchina di Turing (MdT).

La **tesi di Church-Turing** afferma, infatti, che la classe delle funzioni computabili (secondo il concetto intuitivo di algoritmo) coincide con quella delle funzioni calcolabili da una MdT, cioè con la classe delle funzioni Turing-computabili (il termine **funzione computabile** - sinonimi: calcolabile, risolubile - indica un procedimento che trasforma i dati iniziali in dati finali).

Per quel che riguarda la misurazione della risorsa **tempo**, data una macchina di Turing M, si dice che M *opera in tempo* $T(n)$ se dato un input di lunghezza n, la macchina M produce il risultato in $T(n)$ passi.

Per quel che riguarda la misurazione della risorsa **spazio**, data una macchina di Turing M, si dice che M *opera in spazio* $S(n)$ se dato un input di lunghezza n, la macchina M utilizza $S(n)$ celle "temporanee" per effettuare la computazione. Per temporanee si intende che la dimensione dell'input e la dimensione dell'output non vengono conteggiate in $S(n)$.

Classificazione degli algoritmi

I problemi vengono classificati in differenti classi di complessità, in base all'efficienza del migliore algoritmo noto che sia in grado di risolvere quello specifico problema.

Una distinzione informale, ma di grande rilievo, è quella posta tra i cosiddetti problemi facili, di cui si conoscono algoritmi di risoluzione efficienti, e difficili, di cui gli unici algoritmi noti non sono efficienti.

Classificazione degli algoritmi per complessità (dal meno complesso al più complesso):

		notazione O-grande			notazione O-grande
• costante:	1	$O(1)$	• quadratica:	n^2	$O(n^2)$
• logaritmica:	log n	$O(\log n)$	• cubica:	n^3	$O(n^3)$
• lineare:	n	$O(n)$	• esponenziale:	2^n	$O(2^n)$
• linearitmica:	n log n	$O(n \log n)$			

Ordini (di grandezza) **di alcune funzioni comuni** (elencate per magnitudine crescente):

Notazione	**Nome**	**Esempio**
$O(1)$	costante	Determinare se un numero è pari o dispari
$O(\log_2 n)$	logaritmica	Cercare un elemento in una lista ordinata tramite l'algoritmo della ***ricerca binaria***
$O(n)$	lineare	**Ricercare** un elemento in una lista non ordinata
$O(n \log_2 n)$	linearitmica / loglineare	Ordinare una lista tramite ***quick sort***
$O(n^2)$	quadratica	Ordinare una lista tramite ***bubble sort, selection sort***