4.3. Complessità intrinseca di un problema

Un problema di cui si conosce un algoritmo, con la relativa complessità computazionale potrebbe, in futuro, essere risolto algoritmicamente in un altro modo, magari più efficiente (ad esempio con complessità di tempo inferiore)

- Fino a che punto è possibile "migliorare" la soluzione di un problema?
- Fino a che punto possiamo sperare che qualcuno in futuro possa formulare un algoritmo migliore per risolvere il nostro problema?
- Ci sono limiti al di sotto dei quali non è possibile scendere?

Per alcuni problemi la risposta è affermativa: si conosce il limite oltre il quale nessun algoritmo può scendere, per ingegnoso che possa essere.

Si parla di **"lower bound" del problema, ovvero il numero minimo di operazioni che qualsiasi soluzione al problema comporta** e che dipende dalla struttura stessa del problema mentre è indipendente da un qualsiasi algoritmo, anche sofisticatissimo.

Il lower bound di un problema è una proprietà intrinseca dello stesso (spesso facciamo riferimento ad esso chiamandolo complessità intrinseca del problema), ed è un'informazione importantissima per chi sviluppa software.

La conoscenza del lower-bound di un problema ha un'importanza fondamentale: ci permette di non perdere tempo nella ricerca di un algoritmo "troppo" veloce, che non può esistere. Abbiamo così la certezza che nessun "genio" (neanche fra 10.000 anni) potrà mai trovare un algoritmo che sia più veloce del lower-bound del problema.

La dimostrazione di lower-bound è difficile. Bisogna analizzare, ad esempio, la natura del problema, indipendentemente da quella che può essere una possibile strategia di soluzione. Sono pochi i problemi di cui si è dimostrata l'esistenza di lower-bound. Due esempi famosi sono il problema della ricerca di un elemento in una tabella (che affronteremo successivamente) che è equivalente alla ricerca di una parola in un dizionario e il problema dell'ordinamento di una lista di oggetti (siano parole, numeri, ecc…). Nel primo caso il lower-bound è $\log n$, cioè nessun algoritmo (basato su confronto di elementi) può impiegare, nel worst-case, meno di $\log n$ operazioni in presenza di una tabella con n elementi. Nel secondo caso è stato dimostrato che un qualsiasi algoritmo che ordina un insieme di n elementi non può effettuare (nel worst-case) meno di $n \log n$ operazioni elementari.