COMPLESSITÀ COMPUTAZIONALE DEGLI ALGORITMI

Fondamenti di Informatica a.a.2006/07
Prof. V.L. Plantamura
Dott.ssa A. Angelini

Osservazioni

- Ci sono casi in cui non sono verificate le condizioni che sono alla base dell'introduzione delle notazioni asintotiche.
- Ad esempio nel caso in cui è necessario confrontare algoritmi aventi tempi di esecuzione $T(n)$ il cui andamento al limite è uguale (es. sono entrambe $O(n^2)$).
  - In tal caso per stabilire quale algoritmo è più conveniente usare bisogna necessariamente tener conto, in primo luogo delle costanti moltiplicative, e poi dei termini di ordine inferiore.

Osservazioni

Un esempio classico è dato dalla scelta degli algoritmi di ordinamento:

- Si dimostra che il *MergeSort* è un algoritmo che appartiene a $\Theta(nlogn)$, ovvero la sua complessità è sia nel caso migliore che in quello peggiore $\Theta(nlogn)$;
- Il *QuickSort*, viceversa, è un algoritmo $O(n^2)$, la cui complessità nel caso migliore e nel caso medio è però $\Theta(nlogn)$.

Contrariamente a quanto si potrebbe pensare, l'algoritmo di ordinamento più comunemente usato, e che si trova normalmente implementato nelle librerie, è proprio quest'ultimo.

Osservazioni

Questa scelta ha origine da due ordini di considerazioni:

- la prima è di carattere “tecnico”, dovuta al fatto che il *QuickSort* svolge l’ordinamento sul posto, e che quindi ha minore complessità spaziale del *MergeSort,*
- la seconda, invece, è dovuta proprio al fatto che, da un lato la probabilità che per il *QuickSort* si verifichi il caso pessimo è abbastanza remota (e quindi considerarlo come un algoritmo di complessità $\Theta(nlogn)$, pur non essendo formalmente corretto, è una assunzione abbastanza prossima alla realtà) dall’altro che le costanti moltiplicative nel caso del *QuickSort* sono minori rispetto a quelle del *MergeSort*.

Osservazioni

- Un altro caso in cui non è corretto trascurare i termini “nascosti” dalla notazione asintotica è il caso in cui siamo interessati a confrontare il comportamento di due algoritmi per un *prefissato n;*
- In tal caso è possibile, ad esempio, che un algoritmo *A1* di complessità $\Theta(n^3)$ si comporti meglio di un algoritmo $A2 \in \Theta(n^2)$:
  - supponiamo di aver fissato $n = 50$ e che l’algoritmo *A1* abbia un tempo $\boldsymbol{T(n) = (n^3/10)}$ mentre l’algoritmo *A2* abbia $\boldsymbol{T(n) = 10n^2 + 2n + 10}$. In tal caso per *A1* avremo $T(50) = 12500$, mentre per *A2* avremo $T(50) = 25110$.

Complessità dei problemi

- È possibile estendere la nozione di complessità temporale degli algoritmi ai problemi, considerando che dato un problema P, esistono più algoritmi per risolvere P;
- Un Problema P ha complessità $O(f(n))$ se e solo se esiste un algoritmo A per risolvere P di complessità $O(f(n))$.

Esempio

- Il problema di ordinamento. L'algoritmo *bubblesort* ha complessità $O(n^2)$. Possiamo dire che il problema ha complessità $O(n^2)$.
- N.B.: esiste un algoritmo di ordinamento, il *mergesort*, di complessità $O(n\log n)$. Quindi, è anche vero che il problema ordinamento ha complessità $O(n\log n)$.

Complessità dei problemi

- Un Problema P ha complessità $\Omega(g(n))$ se qualunque algoritmo (noto o ignoto) per risolvere P richiede tempo $\Omega(g(n))$;
- È molto difficile stabilire un limite asintotico inferiore, che non sia banale;
- I metodi per stabilirli sono trattati genericamente da: teoria dell'informazione, logica matematica, algebra.

Complessità dei problemi

- Un algoritmo A che risolve un problema P è ottimale se:
  - P ha complessità $\Omega(f(n))$;
  - A ha complessità $O(f(n))$.
- Esempio: Si dimostra che il problema di ordinamento ha complessità $\Omega(n\log n)$. L'algoritmo mergesort ha complessità $O(n\log n)$. L'algoritmo mergesort è ottimale.

Osservazioni

- Dovremmo scrivere che un certa funzione T(n) $\in$ O(n), perché O(*f(n)*) denota la classe di tutte le funzioni maggiorate asintoticamente da f(n);
- Scrivere $T(n) = O(n)$ in alcuni casi è molto utile dal punto di vista notazionale, anche se formalmente scorretto, per poter sommare due notazioni asintotiche, cioè ammettere espressioni del tipo $T(n) = O(n^2) + O(n)$.

Alcune proprietà

- La transitività vale per O, Ω, Θ.
  - Ad esempio $T(n) = O(f(n))$ e $f(n) = O(h(n))$ implicano $T(n) = O(h(n))$
- La riflessività vale per O, Ω, Θ
- La simmetria vale per Θ.
  - Ovvero $T(n) = \Theta(f(n))$ *se e solo se* $f(n) = \Theta(T(n))$
- La simmetria trasposta vale per O e Ω.
  - Ovvero $T(n) = O(f(n))$ *se e solo se* $f(n) = \Omega(T(n))$

Ricapitolando

- Per studiare la complessità temporale di un algoritmo A occorre definire a priori:
  - Un modello di macchina utilizzato;
  - Come misurare la dimensione dell'input;
- Poiché è difficile calcolare esattamente la funzione T(n), ricorriamo alla notazione asintotica.