

Analisi di algoritmi

Determinare la **complessita'** di un algoritmo vuol dire prevedere le risorse che esso richiede.

Risorse → Memoria
 Tempo di esecuzione

Le risorse richieste da un algoritmo dipendono dalle caratteristiche dell' input. La complessita' di un algoritmo deve essere espressa come una funzione della **dimensione dell' input**.

Dimensione dell' input = parametro o insieme di parametri caratterizzanti le proprieta' dell' input

Problema dell' ordinamento

Complessita' = tempo di esecuzione
dimensione input = num. elementi dell' input

Analisi di algoritmi

Per analizzare un algoritmo e' necessario definire un modello di calcolo:

RAM (Random-Access Machine) - le istruzioni sono eseguite una dopo l'altra senza operazioni concorrenti

Il tempo di esecuzione e' determinato dal numero di **operazioni elementari** necessarie per il calcolo dell'output.

Ipotizziamo che:

- Per eseguire ciascuna linea dello pseudocodice e' richiesto un tempo costante c_i ;
- Linee diverse possono richiedere tempi di esecuzione differenti ($c_i \neq c_j$).

Insertion-Sort(A)

For $j \leftarrow 2$ to $\text{length}[A]$	c_1	$n - 1$
do {		
$k \leftarrow A[j]$	c_2	$n - 1$
$i \leftarrow j - 1$	c_3	$n - 1$
while $i > 0$ e $A[i] > k$	c_4	$\sum_{j=2}^n t_j$
do {		
$A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^n (t_j - 1)$
$i \leftarrow i - 1$	c_6	$\sum_{j=2}^n (t_j - 1)$
$A[i+1] \leftarrow k$	c_7	$n - 1$

T_j = numero di volte che, per un fissato j , il test del ciclo while e' eseguito (dipende dall' input).

$T(n)$ = tempo di esecuzione dell'algoritmo.

$$T(n) = c_1(n-1) + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^n t_j +$$

$$+ c_5 \sum_{j=2}^n (t_j - 1) + c_6 \sum_{j=2}^n (t_j - 1) + c_7(n-1)$$

Analisi di Insertion Sort

Espressione generale

$$T(n) = c_1(n-1) + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^n t_j + \\ + c_5 \sum_{j=2}^n (t_j - 1) + c_6 \sum_{j=2}^n (t_j - 1) + c_7(n-1)$$

Caso migliore - $A = \langle a_1, a_2, \dots, a_n \rangle$ ordinata non decrescente

$$t_j \equiv 1 \quad ; \quad j = 2, 3, \dots, n \quad \rightarrow$$

$$\begin{cases} \sum_{j=2}^n t_j = n-1 \\ \sum_{j=2}^n (t_j - 1) = 0 \end{cases} \quad \rightarrow$$

$$T(n) = (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_1 + c_2 + c_3 + c_4 + c_7)$$

Il tempo di esecuzione e' una **funzione lineare** della dimensione dell' input **n**.

Analisi di Insertion Sort

Espressione generale

$$T(n) = c_1(n-1) + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^n t_j + \\ + c_5 \sum_{j=2}^n (t_j - 1) + c_6 \sum_{j=2}^n (t_j - 1) + c_7(n-1)$$

Caso peggiore - $A = \langle a_1, a_2, \dots, a_n \rangle$ ordinata non crescente

$$t_j \equiv j \quad ; \quad j = 2, 3, \dots, n \quad \rightarrow$$

$$\begin{cases} \sum_{j=2}^n t_j = \frac{n(n+1)}{2} - 1 \\ \sum_{j=2}^n (t_j - 1) = \frac{n(n-1)}{2} \end{cases} \quad \rightarrow$$

$$T(n) = (c_4 + c_5 + c_6) \frac{n^2}{2} - \left(c_1 + c_2 + c_3 + c_7 + \frac{c_4 + c_5 + c_6}{2} \right) n - \\ - (c_1 + c_2 + c_3 + c_4 + c_7)$$

Il tempo di esecuzione e' una **funzione quadratica** della dimensione dell' input **n**.

Analisi di Insertion Sort

Espressione generale

$$T(n) = c_1(n-1) + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^n t_j + \\ + c_5 \sum_{j=2}^n (t_j - 1) + c_6 \sum_{j=2}^n (t_j - 1) + c_7(n-1)$$

Caso medio - $A = \langle a_1, a_2, \dots, a_n \rangle$ **casuale**

In media $t_j = j/2$; $j = 2, 3, \dots, n$ \rightarrow

Il tempo di esecuzione e' una **funzione quadratica** della dimensione dell' input **n**.

N.B.

Nell'analisi degli algoritmi si considera in genere il caso peggiore. Cio' poiche':

- Si ottiene in tal modo un limite superiore sui tempi di esecuzione;
- In genere: **(medio) \approx (peggiore)**

→ L' algoritmo di Insertion Sort ha complessita' quadratica in n:

$$T(n) = a n^2 + b n + c$$

Cio' implica che:

-Asintoticamente (cioe' per n sufficientemente grandi) il tempo di esecuzione dell'algoritmo e' proporzionale ad n^2 .

Perche' e' importante chiedersi come cresce $T(n)$?
Ipotizziamo di risolvere un problema computazionale mediante due diversi algoritmi:

$$T_1(n) = a n^2 + b n + c \rightarrow a n^2$$

$$T_2(n) = b' n + c' \rightarrow b' n$$

N.B.

- Il valore delle costanti dipende dal processore.
- La dipendenza da n dipende solamente dall'algoritmo utilizzato.

Indipendentemente dalle costanti a e b', esiste $n^* = (b' / a)$ tale che:

$$\text{per ogni } n > n^* \quad T_2(n) > T_1(n)$$