Notazione asintotica

Sebbene si possa talvolta determinare il tempo esatto di esecuzione di un algoritmo, l'estrema precisione non giustifica lo sforzo del calcolo; infatti, per input sufficientemente grandi, le costanti moltiplicative e i termini di ordine più basso di tale tempo sono trascurabili rispetto agli effetti della dimensione stessa dell'input

Per input sufficientemente grandi da rendere rilevante solo l'ordine di grandezza del tempo di esecuzione, si studia solo l'efficienza asintotica, definita in termini di funzioni il cui dominio è l'insieme dei n° naturali (zero incluso), usando alcuni tipi di notazione asintotica:

- notazione Θ
- notazione O
- notazione Ω
- notazione o
- notazione ω

Notazione Q

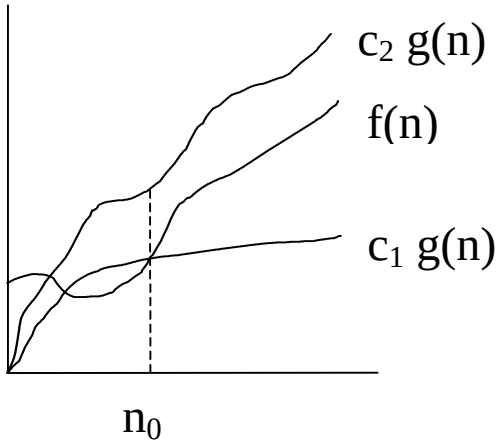
$\Theta(g(n)) = \{f(n) \mid \exists\, c_1>0, \exists\, c_2>0, \exists\, n_0>0, \forall\, n \geq n_0,\ 0 \leq c_1\, g(n) \leq f(n) \leq c_2\, g(n)\}$

dove sia g(n), sia f(n) sono funzioni asintoticamente non negative (cioè non negative per $\forall$n sufficientemente grande)

Per indicare che f(n) è membro di $\Theta(g(n))$ si scrive

$f(n) \in \Theta(g(n))$ oppure
$f(n) = \Theta(g(n))$ (uso improprio ma diffusissimo del segno di uguaglianza)

Si dice che g(n) è un <u>limite asintotico stretto</u> per $\forall$ $f(n) \in \Theta(g(n))$



$f(n) = \Theta(g(n))$

Notazione Θ (cont.)

La notazione Θ limita asintoticamente una funzione da sopra e da sotto

Intuitivamente, i termini di ordine più basso di una funzione asintoticamente positiva (ovvero positiva per $\forall n$ sufficientemente grande) possono essere ignorati nella determinazione del limite asintotico stretto perché essi sono trascurabili per n grande (una piccola frazione del termine di ordine più alto domina i termini di ordine più basso)

L'assegnamento a c_1 di un valore leggermente più piccolo del coefficiente del termine di ordine più alto e a c_2 di un valore leggermente più grande consente che le disuguaglianze della definizione di Θ siano soddisfatte

Qualunque funzione costante si può esprimere come $\Theta(n^0)$ o, più comunemente, $\Theta(1)$, anche se quest'ultima notazione è impropria perché non evidenzia quale variabile tenda a ∞

Notazione O

Si legge "o grande di *g* di *n*"

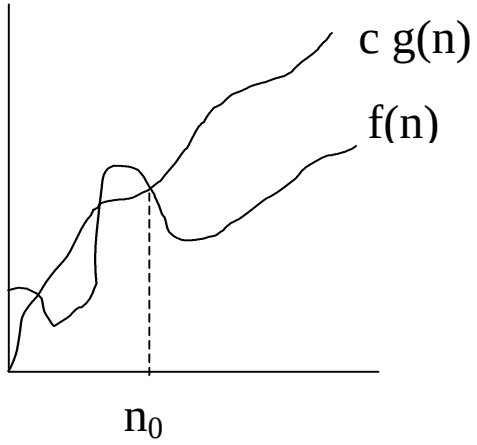
$O(g(n)) = \{f(n) \mid \exists\, c>0, \exists\, n_0>0, \forall\, n \geq n_0,\ 0 \leq f(n) \leq c\, g(n)\}$

dove sia g(n), sia f(n) sono funzioni asintoticamente non negative

Per indicare che f(n) è membro di O(g(n)) si scrive f(n) = O(g(n))

$f(n) = \Theta(g(n)) \rightarrow f(n) = O(g(n))$
o, detto altrimenti, $\Theta(g(n)) \subseteq O(g(n))$

Si dice che g(n) è un <u>limite asintotico superiore</u> per $\forall$ f(n) = O(g(n))



f(n) = O(g(n))

Notazione Ω

Si legge "omega grande di *g* di *n*"

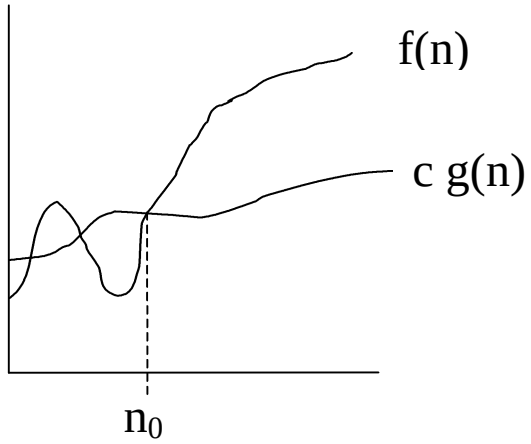
$\Omega(g(n)) = \{f(n) \mid \exists\, c>0, \exists\, n_0>0, \forall\, n \geq n_0,\ 0 \leq c\, g(n) \leq f(n)\}$

dove sia g(n), sia f(n) sono funzioni asintoticamente non negative

Per indicare che f(n) è membro di $\Omega(g(n))$ si scrive $f(n) = \Omega(g(n))$

$f(n) = \Theta(g(n)) \rightarrow f(n) = \Omega(g(n))$
o, detto altrimenti, $\Theta(g(n)) \subseteq \Omega(g(n))$

Si dice che g(n) è un <u>limite asintotico inferiore</u> per $\forall$ $f(n) = \Omega(g(n))$



$f(n) = \Omega(g(n))$

Notazione Q**, O e** W

Teorema

Per ogni coppia di funzioni f(n) e g(n), $f(n) = \Theta(g(n))$ sse $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$.

Notazione Q, O e W (cont.)

$T^p(n) = O(g(n)) \rightarrow T(n) = O(g(n))$
un limite asintotico superiore del tempo di esecuzione di un algoritmo nel caso peggiore è anche un limite asintotico superiore per il tempo di esecuzione su input arbitrari
Ad es. per INSERTION-SORT $T^p(n) = O(n^2) \rightarrow T(n) = O(n^2)$

$T^p(n) = \Theta(g(n)) \not\rightarrow T(n) = \Theta(g(n))$
un limite asintotico stretto del tempo di esecuzione di un algoritmo nel caso peggiore NON è un limite asintotico stretto per il tempo di esecuzione su input arbitrari
Ad es. per INSERTION-SORT $T^p(n) = \Theta(n^2) \not\rightarrow T(n) = \Theta(n^2)$, infatti, quando l'input è già ordinato, $T(n) = \Theta(n)$

Notazione Q, O e W (cont.)

$T^{best}(n) = \Omega(g(n)) \rightarrow T(n) = \Omega(g(n))$
un limite asintotico inferiore del tempo di esecuzione di un algoritmo nel caso migliore è anche un limite asintotico inferiore per il tempo di esecuzione su input arbitrari
Ad es. per INSERTION-SORT $T^{best}(n) = \Omega(n) \rightarrow T(n) = \Omega(n)$

Quindi T(n) di INSERTION-SORT

- è compreso tra $\Omega(n)$ e $O(n^2)$, cioè cade ovunque fra una funzione lineare e una quadratica
- nel caso peggiore è $\Omega(n^2)$

Notazione asintotica nelle equazioni

Quando una notazione asintotica si trova

- da sola sul lato dx di un'equazione, il segno '=' indica appartenenza
  Ad es. $n = O(n^2)$ significa $n \in O(n^2)$
- in ogni altra posizione entro una formula indica una anonima funzione (singola) appartenente all'insieme rappresentato dalla notazione asintotica
  Ad es.
  - $T(n) = 2T(n/2) + \Theta(n)$ ← una generica funzione lineare
  - $\sum_{i=1}^{n} O(i)$ significa dove $\sum_{i=1}^{n} f(i)$ f(i) = O(i)

Notazione asintotica nelle equazioni (cont.)

Si possono anche eguagliare due espressioni, entrambe contenenti una o più notazioni asintotiche; in tal caso il lato dx dell'equazione fornisce un livello di dettaglio più grossolano del lato sx
Ad es. in

$2n^2 + \Theta(n) = \Theta(n^2)$

indipendentemente da come si è scelta la funzione anonima sulla sx, vi è un modo per scegliere la funzione anonima sulla dx per rendere valida l'equazione

Notazione o

Si legge "o piccolo di *g* di *n*"

$o(g(n)) = \{f(n) \mid \forall\, c>0, \exists\, n_0>0, \forall\, n \geq n_0,\ 0 \leq f(n) < c\, g(n)\}$

dove sia g(n), sia f(n) sono funzioni asintoticamente non negative

Per indicare che f(n) è membro di o(g(n)) si scrive f(n) = o(g(n))

$f(n) = o(g(n)) \rightarrow f(n) = O(g(n))$
ovvero, detto altrimenti, $o(g(n)) \subseteq O(g(n))$

Si dice che g(n) è un <u>limite superiore non asintoticamente stretto</u> per
$\forall$ f(n) = o(g(n))
Ciò si verifica sse f(n) diventa trascurabile rispetto a g(n) per n che tende a ∞, cioè

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

(se il limite ∃)

Notazione w

Si legge “omega piccolo di *g* di *n*”

$\omega(g(n)) = \{f(n) \mid \forall\ c>0, \exists\ n_0>0, \forall\ n \geq n_0,\ 0 \leq c\ g(n) < f(n)\}$

dove sia g(n), sia f(n) sono funzioni asintoticamente non negative

Per indicare che f(n) è membro di $\omega(g(n))$ si scrive $f(n) = \omega(g(n))$

$f(n) = \omega(g(n)) \rightarrow f(n) = \Omega(g(n))$
ovvero, detto altrimenti, $\omega(g(n)) \subseteq \Omega(g(n))$

Si dice che g(n) è un limite inferiore non asintoticamente stretto per
$\forall\ f(n) = \omega(g(n))$
Ciò si verifica sse f(n) diventa arbitrariamente grande rispetto a g(n) per n che tende a ∞, cioè

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

(se il limite ∃)

Proprietà delle notazioni asintotiche

Transitività

$f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) \rightarrow f(n) = \Theta(h(n))$
$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \rightarrow f(n) = O(h(n))$
$f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n)) \rightarrow f(n) = \Omega(h(n))$
$f(n) = o(g(n)) \wedge g(n) = o(h(n)) \rightarrow f(n) = o(h(n))$
$f(n) = \omega(g(n)) \wedge g(n) = \omega(h(n)) \rightarrow f(n) = \omega(h(n))$

Riflessività

$f(n) = \Theta(f(n))$
$f(n) = O(f(n))$
$f(n) = \Omega(f(n))$

Simmetria

$f(n) = \Theta(g(n)) \leftrightarrow g(n) = \Theta(f(n))$

Simmetria trasposta

$f(n) = O(g(n)) \leftrightarrow g(n) = \Omega(f(n))$
$f(n) = o(g(n)) \leftrightarrow g(n) = \omega(f(n))$

Analogie e differenze fra notazioni asintotiche e numeri reali

Confronto fra notazioni asintotiche	Analogo confronto fra numeri reali
$a(n) = O(b(n))$	$a \leq b$
$a(n) = \Omega(b(n))$	$a \geq b$
$a(n) = \Theta(b(n))$	$a = b$
$a(n) = o(b(n))$	$a < b$
$a(n) = \omega(b(n))$	$a > b$

Differenza: due n° reali possono sempre essere confrontati mentre esistono funzioni asintoticamente non confrontabili (o incommensurabili), ad es.

- n e $n^{1+sin n}$
- $f(n) \begin{cases} n \text{ per } n \text{ dispari} \\ n^2 \text{ per } n \text{ pari} \end{cases}$ e $g(n) = \begin{cases} n \text{ per } n \text{ pari} \\ n^2 \text{ per } n \text{ dispari} \end{cases}$

Ricorrenze

Ricorrenza = equazione o disuguaglianza che descrive una funzione in termini del suo valore su input più piccoli

Consente di descrivere il tempo di esecuzione di un algoritmo che contiene una chiamata ricorsiva a se stesso

Metodi risolutivi delle ricorrenze

- metodo di sostituzione
- metodo iterativo
- metodo principale

Dettagli tecnici

Nell'uso e risoluzione di ricorrenze che rappresentano tempi di esecuzione di algoritmi sono spesso mascherati alcuni aspetti:

- si tace l'ipotesi che le funzioni abbiano argomenti interi; ad es. per MERGE-SORT si scrive

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1, \\ 2T(n/2) + \Theta(n) & \text{se } n > 1 \end{cases}$$

anziché (come sarebbe corretto)

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1, \\ 2T(\lceil n/2 \rceil) + \Theta(\lfloor n \rfloor) & \text{se } n > 1 \end{cases}$$

Si dimostra che l'omissione delle funzioni base e tetto non influisce sul comportamento asintotico della ricorrenza

- si omettono le condizioni al contorno; in genere si assume tacitamente $T(n) = \Theta(1)$ per n piccolo, cioè ogni condizione al contorno è $\Theta(1)$; per es. per MERGE-SORT si scrive semplicemente $T(n) = 2T(n/2) + \Theta(n)$

Metodo di sostituzione

Si tenta un limite asintotico (secondo la definizione dello stesso) e quindi si usa l'induzione matematica per determinare i valori delle costanti e provare se il tentativo è corretto

Euristiche

- Se una ricorrenza è simile ad altre già viste, tentare una soluzione simile
- Provare limiti laschi della ricorrenza e successivamente ridurre il grado di incertezza

Metodo iterativo

Si sviluppa la ricorrenza come somma di termini dipendenti solo da n e dalle condizioni iniziali, quindi si applicano le tecniche per valutare le sommatorie per fornire limiti alla soluzione

Operativamente è necessario stabilire

- il n° di volte che la ricorrenza deve essere iterata per raggiungere le condizioni al contorno
- la somma dei termini generati da tutte le iterazioni

Alberi di ricorsione

Visualizzazione grafica del metodo iterativo, utile soprattutto quando la ricorrenza descrive un algoritmo divide-et-impera

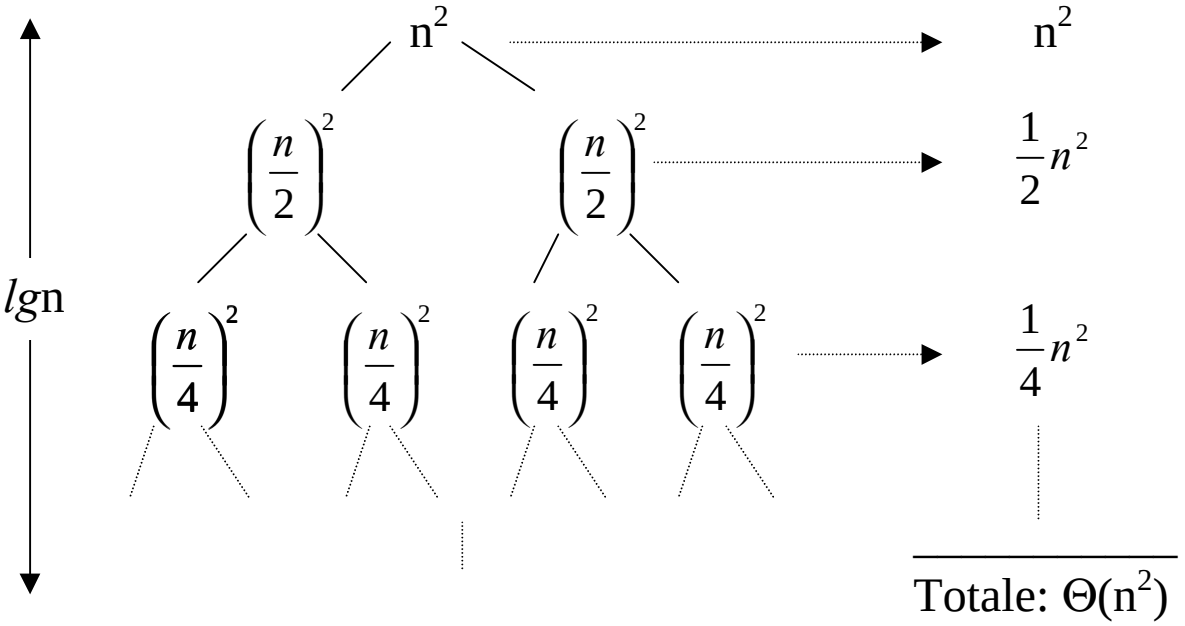
$T(n) = aT(n/b) + D(n) + C(n)$

impera divide combina

Esempio

$T(n) = 2T(n/2) + n^2$

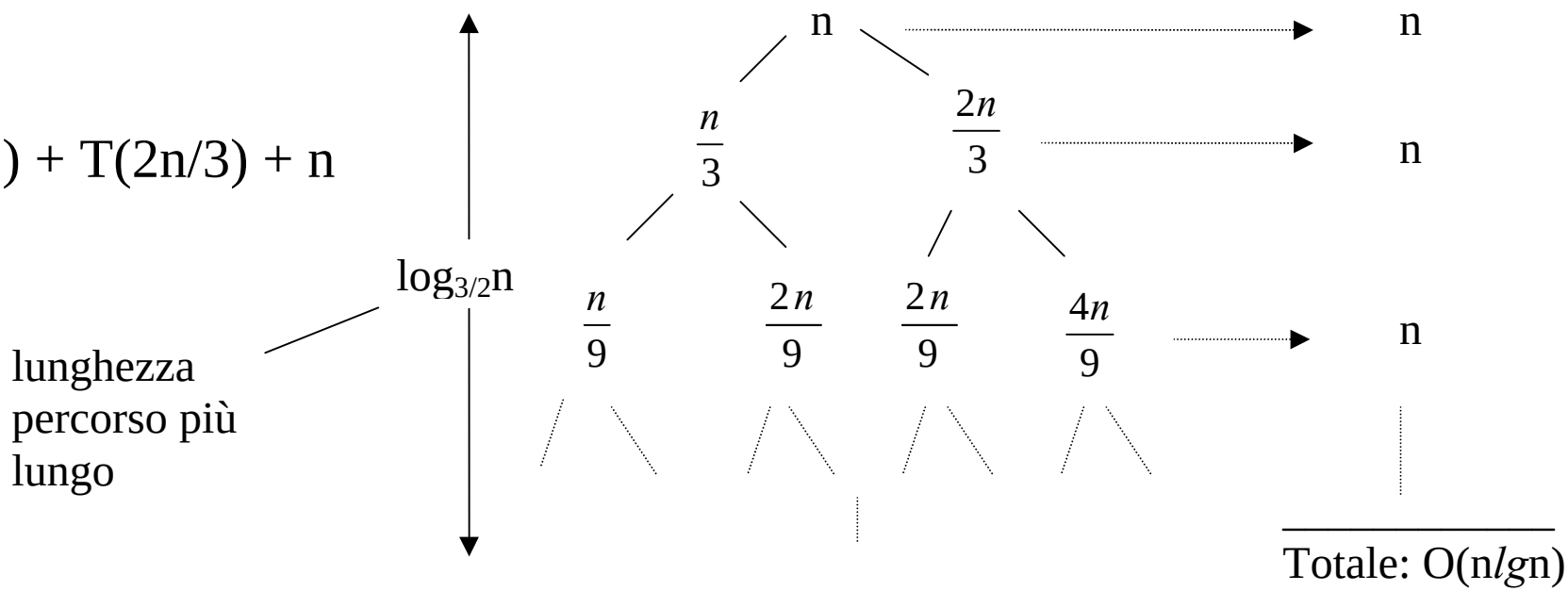
(se n è una potenza di 2)



Alberi di ricorsione (cont.)

Esempio

$T(n) = T(n/3) + T(2n/3) + n$



N.B. All'interno di un'espressione O non è necessario specificare la base dei logaritmi. Infatti $\log_a n$ e $\log_b n$ differiscono solo per una costante → le funzioni $\log_x n$ sono dunque O una dell'altra → per la proprietà transitiva possono essere sostituite l'una con l'altra entro l'espressione O

Metodo principale

Fornisce la soluzione di ricorrenze della forma

$T(n) = a\,T(n/b) + f(n)$

(tempo di esecuzione di un algoritmo ricorsivo che divide un problema di dimensione *n* in *a* sottoproblemi, ciascuno di dimensione *n/b*, impiegando un tempo f(n) per dividere il problema e comporne la soluzione)

dove

- a = costante, b= costante, $a \geq 1$ e $b > 1$
- f(n) = funzione asintoticamente positiva

Metodo principale (cont.)

Teorema principale

Siano $a \geq 1$ e $b > 1$ costanti e $f(n)$ una funzione; $T(n)$ sia definito sugli interi non negativi dalla ricorrenza:

$T(n) = a\,T(n/b) + f(n)$

dove n/b rappresenta $\lceil n/b \rceil$ o $\lfloor n/b \rfloor$.

1. Se $f(n) = O(n^{\log_b a - \varepsilon})$ per qualche costante $\varepsilon > 0$, allora $T(n) = \Theta(n^{\log_b a})$.
2. Se $f(n) = \Theta(n^{\log_b a})$, allora $T(n) = \Theta(n^{\log_b a}\, lg\, n)$.
3. Se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ per qualche costante $\varepsilon > 0$, e se $af(n/b) < cf(n)$ (condizione di regolarità) per qualche costante $c < 1$ e per ogni n sufficientemente grande, allora $T(n) = \Theta(f(n))$.

Metodo principale (cont.)

Operativamente, per applicare il teorema principale,

a) si confronta f(n) con $n^{\log_b a}$ (la più grande delle due determina la soluzione)

b)

- se la più grande è $n^{\log_b a}$ (caso 1) e f(n) è polinomialmente più piccola di $n^{\log_b a}$ per un fattore n^{ε} per qualche costante $\varepsilon > 0$, la soluzione è $T(n) = \Theta(n^{\log_b a})$
- se la più grande è f(n) (caso 3) e lo è polinomialmente e soddisfa la condizione di regolarità, la soluzione è $T(n) = \Theta(f(n))$
- se hanno lo stesso ordine di grandezza (caso 2), si moltiplica tale ordine per un fattore logaritmico, cioè $T(n) = \Theta(n^{\log_b a} \, lg n) = \Theta(f(n) \, lg n)$

N.B. I tre casi non comprendono tutte le possibilità per f(n)

Metodo principale (cont.)

Teorema principale: estensione

2'. Se $f(n) = \Theta(n^{\log_b a} \lg^k n)$, con $k \geq 0$, allora $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.

(sussume il caso 2, che si ottiene per $k = 0$)