Progetto e analisi di algoritmi

Roberto Cordone

DTI - Università degli Studi di Milano

Polo Didattico e di Ricerca di Crema

Tel. 0373 / 898**089**

E-mail: **`cordone@dti.unimi.it`**

Ricevimento: **su appuntamento**

Web page: **`http://www.dti.unimi.it/~cordone`**

Lezioni: **Martedì dalle 11.00 alle 13.00**

Giovedì dalle 11.00 alle 13.00

Equazioni ricorrenti

Legano il valore di una funzione $f : \mathbb{N} \to \mathbb{R}$ nel punto n ai suoi valori in punti precedenti $\bar{n}, n_1, \ldots, n_k < n$

$$f(n) = \begin{cases} c & n = \bar{n} \\ \Phi\left(f(n_1), \ldots, f(n_k)\right) & n > n_i \; (i = 1, \ldots, k) \end{cases}$$

Un'applicazione importante è esprimere la complessità temporale di un algoritmo ricorsivo; ad esempio

$$T(n) = \begin{cases} k & n = 1 \\ D(n) + aT\left(\frac{n}{b}\right) + C(n) & n > 1 \end{cases}$$

Algoritmi ricorsivi

$$T(n) = \begin{cases} k & n = 1 \\ D(n) + aT\left(\frac{n}{b}\right) + C(n) & n > 1 \end{cases}$$

- Divide $D(n)$: suddividi il problema
- Impera $aT(n/b)$: risolvi a sottoproblemi di dimensione n/b
- Combina $C(n)$: combina le soluzioni dei sottoproblemi in quella complessiva

Se i sottoproblemi sono di dimensione diversa, il termine $aT(n/b)$ si complica

Esempio: MergeSort

MergeSort(A, s, d)

If $s < d$ **then** { Se $s \geq d$, non si fa nulla (caso base) }

$m = \left\lfloor \frac{s+d}{2} \right\rfloor$ { Divide }

MergeSort(A, s, m); { Impera (1) }

MergeSort(A, $m+1$, d); { Impera (2) }

Merge(A, s, m, d); { Combina }

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ \Theta(1) + 2T\left(\frac{n}{2}\right) + \Theta(n) & n > 1 \end{cases}$$

Esempio: Fattoriale

Fattoriale(n)

If $n \leq 1$

then **Return** 1; { caso base }

else $a = n - 1$ { Divide }

b := Fattoriale(a); { Impera }

Return $n \cdot b$; { Combina }

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ \Theta(1) + T(n-1) + \Theta(1) & n > 1 \end{cases}$$

Tecniche risolutive

La via maestra per risolvere le equazioni ricorrenti è il metodo di sostituzione: ipotizzare una soluzione e verificarla per induzione matematica

Esistono anche

- metodo iterativo: sviluppare l'equazione sino al caso base, esplicitando T (il metodo suggerisce soluzioni, che andrebbero verificate per induzione, ma talvolta vengono direttamente accettate)
- metodo principale: ricondursi all'ampia classe di equazioni risolte dal *master theorem* (se possibile!)

Obiettivo

A seconda del contesto, può interessare

- la risoluzione esatta dell'equazione:
  determinare l'espressione di $f(\cdot)$ per ogni $n \in \mathbb{N}$
- la complessità asintotica della soluzione $f(\cdot)$:
  determinare un limite superiore e/o inferiore, valido per ogni $n \geq n_0$

Nel caso degli algoritmi ricorsivi, in genere è sufficiente il secondo risultato, che permette alcune semplificazioni

Metodo di sostituzione

- ipotizzare la forma della soluzione
- dimostrare l'ipotesi per induzione matematica

$$T(n) = \begin{cases} 1 & \text{per } n = 1 \\ 4T\left(\frac{n}{2}\right) - \frac{n}{2} & \text{per } n > 1 \end{cases} \qquad \text{(potenze di 2)}$$

Dimostriamo che $T(n) = G(n) = \frac{1}{2}n^2 + \frac{1}{2}n$

T1 $T(1) = G(1)$

T2 se $T(i) = G(i)$ per $i = 1, \ldots, n-1$, allora $T(n) = G(n)$

Dimostrare il caso base

Ipotesi:

1. $T(n) = \begin{cases} 1 & \text{per } n = 1 \\ 4T\left(\frac{n}{2}\right) - \frac{n}{2} & \text{per } n > 1 \end{cases}$
2. $G(n) = \frac{1}{2}n^2 + \frac{1}{2}n$

Tesi 1: $T(1) = G(1)$

Dimostrazione:

$$G(1) = \frac{1}{2}\,1^2 + \frac{1}{2}\,1 = 1 = T(1)$$

Dimostrare il passo induttivo (1)

Ipotesi:

1. $T(n) = \begin{cases} 1 & \text{per } n = 1 \\ 4T\left(\frac{n}{2}\right) - \frac{n}{2} & \text{per } n > 1 \end{cases}$
2. $G(n) = \frac{1}{2}n^2 + \frac{1}{2}n$
3. $n > 1$
4. $T(i) = G(i)$ per $i = 1, \ldots, n-1$

Tesi 2: $T(n) = G(n)$

Dimostrare il passo induttivo (2)

Dimostrazione: Per $n > 1$ (Hp. 3) è $n/2 \leq n - 1$ (1)

Hp. 4: $T\left(\frac{n}{2}\right) = G\left(\frac{n}{2}\right)$ (2)

Hp. 1: $T(n) = 4T\left(\frac{n}{2}\right) - \frac{n}{2}$

(2): $T(n) = 4G\left(\frac{n}{2}\right) - \frac{n}{2}$

Hp. 2: $T(n) = 4\left[\frac{1}{2}\left(\frac{n}{2}\right)^2 + \frac{1}{2}\left(\frac{n}{2}\right)\right] - \frac{n}{2}$

$$T(n) = 4\frac{n^2}{8} + n - \frac{n}{2} = \frac{1}{2}n^2 + \frac{1}{2}n = G(n)$$

Per induzione forte, $T(n) = G(n)$ per ogni $n \in \mathbb{N}$

Come si indovina $T(n)$*?*

Un modo è indovinare la forma a meno di qualche coefficiente e poi determinare questi ultimi

$$T(n) = \begin{cases} 1 & \text{per } n = 1 \\ 4T\left(\frac{n}{2}\right) - \frac{n}{2} & \text{per } n > 1 \end{cases}$$

Ipotizziamo che sia $\exists a, b, c : G(n) = an^2 + bn + c$

Per dimostrare il caso base basta imporre

$$T(1) = G(1) \Rightarrow 1 = a + b + c$$

Come si indovina $T(n)$?

Passo induttivo: si ipotizza $T(i) = G(i)$ per $i = 1, \ldots, n-1$

Poiché $n > 1$, è $n/2 \leq n - 1 \Rightarrow T\left(\frac{n}{2}\right) = G\left(\frac{n}{2}\right) \Rightarrow$

$$\Rightarrow T(n) \;=\; 4G\left(\frac{n}{2}\right) - \frac{n}{2} = 4\left[a\left(\frac{n}{2}\right)^2 + b\frac{n}{2} + c\right] - \frac{n}{2} \Rightarrow$$

$$\Rightarrow T(n) \;=\; an^2 + \left(2b - \frac{1}{2}\right)n + 4c$$

Vogliamo dimostrare che $T(n) = G(n) = an^2 + bn + c$, $\forall n > 1$

L'unico modo è uguagliare i coefficienti di ogni potenza di n

$$\Rightarrow a = a, \quad 2b - \frac{1}{2} = b, \quad 4c = c \qquad \Rightarrow \qquad b = \frac{1}{2}, \quad c = 0$$

che insieme ad $a + b + c = 1$ implicano $a = \frac{1}{2}$

Spesso non funziona

$$T(n) = \begin{cases} 1 & \text{per } n = 1 \\ 2T\left(\frac{n}{3}\right) + n^2 & \text{per } n > 1 \end{cases}$$

Se ipotizziamo $T(n) \equiv G(n) = an^2 + bn + c$ si ottiene

$$\begin{cases} a + b + c = 1 \\ 1 + \dfrac{2}{9}a = a \\ \dfrac{2}{3}b = b \\ 2c = c \end{cases} \Rightarrow \begin{cases} a + b + c = 1 \\ a = \dfrac{9}{7} \\ b = 0 \\ c = 0 \end{cases}$$

che è impossibile!

Spesso non funziona

La famiglia di funzioni ipotizzata non contiene la soluzione dell'equazione ricorrente

- la soluzione comprende termini diversi da quelli usati (come $n^{7/4}$ o $n \log n$, ecc. . .)
- la soluzione non ha un'espressione analitica (ci sono molte meno funzioni che algoritmi!)

Approssimazioni asintotiche

Un altro modo è procedere per approssimazioni successive dall'alto e dal basso: individuare per tentativi funzioni che crescono più rapidamente e meno rapidamente di $T(n)$

$$T(n) = \begin{cases} a & \text{per } n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & \text{per } n > 1 \end{cases}$$

dove $a > 0$ e $b > 0$ sono costanti assegnate

Verifichiamo che $T(n) \in O(n^2)$, cioè che

$$\exists n_0 \in \mathbb{N}, c > 0 : \quad T(n) \leq cn^2, \quad \forall n \geq n_0$$

Dimostrazione di complessità

Dimostrare $T(n) \in O(n^2)$ per induzione richiede

T1 $\exists n_0 \in \mathbb{N}, c > 0 : T(n_0) \leq c n_0^2$

T2 $\exists n_0 \in \mathbb{N}, c > 0 :$ se $T(i) \leq c i^2$ per $i = n_0, \ldots, n-1$, allora $T(n) \leq c n^2$

Quando si cerca la complessità asintotica di $T(n)$ anziché l'espressione esatta, la base dell'induzione (n_0) può differire dalla base dell'equazione ricorrente ($\bar{n} = 1$)

$T(n) \in O(n^2)$ (caso base)

Ipotesi:

1. $T(n) = \begin{cases} a & \text{per } n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & \text{per } n > 1 \end{cases}$

   con $a > 0$ e $b > 0$ costanti assegnate

2. $n_0 = \bar{n}_0 \in \mathbb{N}$ (valore scelto da noi)

3. $c = \bar{c} > 0$ (valore scelto da noi)

Tesi 1: $T(n_0) \leq cn_0^2$

Dimostrazione: Supponiamo per semplicità $\bar{n}_0 = 1$
La tesi $T(n_0) = a \leq c \cdot 1^2$ è verificata per $\bar{c} \geq a$

$T(n) \in O(n^2)$ *(passo induttivo)*

Ipotesi:

1. $T(n) = \begin{cases} a & \text{per } n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & \text{per } n > 1 \end{cases}$
   con $a > 0$ e $b > 0$ costanti assegnate
2. $n_0 = \bar{n}_0 \in \mathbb{N}$ (valore scelto da noi)
3. $c = \bar{c} > 0$ (valore scelto da noi)
4. $n > n_0$ (valore generico)
5. $T(i) \leq ci^2$ per $i = n_0, \ldots, n-1$

Tesi 2: $T(n) \leq cn^2$

$T(n) \in O\left(n^2\right)$ *(passo induttivo)*

Dimostrazione: Avendo posto $\bar{n}_0 = 1$, è $n > 1$ (Hp. 2 e 4) (1)

(1): $n/2 \leq n - 1$ (2)

Hp. 5: $T\left(\frac{n}{2}\right) \leq c\left(\frac{n}{2}\right)^2$ (3)

Hp. 1: $T(n) = 2T\left(\frac{n}{2}\right) + bn \leq$

(3): $\leq 2c\left(\frac{n}{2}\right)^2 + bn < \left(\frac{c}{2} + b\right) n^2$

che è $\leq cn^2$ per $c \geq 2b$

Per dimostrare il caso base si è già imposto $\bar{c} \geq a$

Posto $\bar{c} = \max(a, 2b)$, segue la tesi $T(n) \in O\left(n^2\right)$

$T(n) \in O(n)$? (caso base)

La maggiorazione $2c\left(\frac{n}{2}\right)^2 + bn < \left(\frac{c}{2} + b\right)n^2$ è piuttosto forte (da bn a bn^2!) e suggerisce una complessità inferiore: $O(n)$?

Ipotesi:

1. $T(n) = \begin{cases} a & \text{per } n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & \text{per } n > 1 \end{cases}$ con $a > 0$ e $b > 0$
2. $n_0 = \bar{n}_0 \in \mathbb{N}$ (valore scelto da noi)
3. $c = \bar{c} > 0$ (valore scelto da noi)

Tesi 1: $T(n_0) \leq cn_0$

Dimostrazione: Poniamo ancora per semplicità $\bar{n}_0 = 1$

$T(n_0) = a \leq c \cdot 1$ per ogni $\bar{c} \geq a$

$T(n) \in O(n)$? (passo induttivo)

Ipotesi:

1. $T(n) = \begin{cases} a & \text{per } n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & \text{per } n > 1 \end{cases}$

   con $a > 0$ e $b > 0$ costanti assegnate
2. $n_0 = \bar{n}_0 \in \mathbb{N}$ (valore scelto da noi)
3. $c = \bar{c} > 0$ (valore scelto da noi)
4. $n > n_0$ (valore generico)
5. $T(i) \leq ci$ per $i = n_0, \ldots, n-1$

Tesi 2: $T(n) \leq cn$

$T(n) \in O(n)$? (passo induttivo)

Dimostrazione: Avendo posto $\bar{n}_0 = 1$, è $n > 1$ (Hp. 2 e 4) (1)

$$\text{(1):} \quad n/2 \leq n - 1 \tag{2}$$

$$\text{Hp. 5:} \quad T\left(\frac{n}{2}\right) \leq c\left(\frac{n}{2}\right) \tag{3}$$

$$\text{Hp. 1:} \quad T(n) = 2T\left(\frac{n}{2}\right) + bn \leq$$

$$\text{(3):} \quad \leq 2c\left(\frac{n}{2}\right) + bn = (c+b)\,n \not\leq \mathbf{cn}$$

Per $b > 0$, non c'è modo di rendere $T(n) \leq cn$

Potremmo cambiare n_0, ma rimarrebbe $(c+b)\,n \not\leq cn$

Probabilmente $T(n) \notin O(n)$ (non è una dimostrazione)

$T(n) \in \Theta(n \log n)$

Da qualche parte fra $O(n)$ e $O(n^2)$: $T(n) \in \Theta(n \log n)$?

Ipotesi:

1. $T(n) = \begin{cases} a & \text{per } n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & \text{per } n > 1 \end{cases}$

   con $a > 0$ e $b > 0$ costanti assegnate

2. $n_0 = \bar{n}_0 \in \mathbb{N}$ (valore scelto da noi)
3. $c_1 = \bar{c}_1 > 0$ e $c_2 = \bar{c}_2 > 0$ (valore scelto da noi)
4. $n > n_0$ (valore generico)
5. $T(i) \leq ci$ per $i = n_0, \ldots, n-1$

Tesi 2: $c_1 n \log_2 n \leq T(n) \leq c_2 n \log_2 n$

$T(n) \in \Theta(n \log n)$

Dimostrazione: Posto $\bar{n}_0 = 1$, è $n/2 \leq n - 1$ (Hp. 2 e 4)

Hp. 5: $$c_1 \frac{n}{2} \log_2 \frac{n}{2} \leq T\left(\frac{n}{2}\right) \leq c_2 \frac{n}{2} \log_2 \frac{n}{2}$$

$$2c_1 \frac{n}{2} \log_2 \frac{n}{2} + bn \leq 2T\left(\frac{n}{2}\right) + bn \leq 2c_2 \frac{n}{2} \log_2 \frac{n}{2} + bn \qquad (1)$$

Hp. 1: $$2c_1 \frac{n}{2} \log_2 \frac{n}{2} + bn \leq T(n) \leq 2c_2 \frac{n}{2} \log_2 \frac{n}{2} + bn$$

(1): $$c_1 n \log_2 n - c_1 n + bn \leq T(n) \leq c_2 n \log_2 n - c_2 n + bn$$

Se poniamo $\bar{c}_1 \leq b$ e $\bar{c}_2 \geq b$, concludiamo che

$$c_1 n \log_2 n \leq T(n) \leq c_2 n \log_2 n$$

che è la tesi

$T(n) \in \Theta(n \log n)$

Il caso base è problematico

$$\exists n_0 \in \mathbb{N}, c_1, c_2 > 0 : c_1 n_0 \log_2 n_0 \leq T(n_0) \leq c_2 n_0 \log_2 n_0$$

Con $n_0 = 1$: $c_1 \ 1 \log_2 1 \leq a \leq c_2 \ 1 \log_2 1 \Rightarrow a = 0$,
ma l'ipotesi è che $a > 0$!

Possiamo modificare n_0 (riverificando il passo induttivo):

$$n_0 = 2 \Rightarrow T(n_0) = 2a + 2b \Rightarrow$$

$$\Rightarrow c_1 \ 2 \log_2 2 \leq 2a + 2b \leq c_2 \ 2 \log_2 2 \Rightarrow$$

$$\Rightarrow c_1 \leq a + b \leq c_2$$

Per la tesi basta

$$n_0 = 2 \quad c_1 \leq \min(b, a+b) \quad c_2 \geq \max(b, a+b)$$

Come trovare l'espressione giusta?

Non ci sono regole certe, ma ci sono buone tecniche euristiche

1. tentare famiglie di funzioni dipendenti da parametri e determinare questi ultimi
2. tentare limiti superiori e inferiori via via più stringenti
3. tentare con espressioni simili a casi risolti

   $T(n) = 2T\left(\frac{n}{2}+17\right)+n$ somiglia a $T(n) = 2T\left(\frac{n}{2}\right)+n$

   e la soluzione (asintotica) di entrambe le equazioni è

$$T(n) \in \Theta(n \log n)$$

Come trovare l'espressione giusta?

4. cambiare variabili

$$T\left(n\right) = 2T\left(\sqrt{n}\right) + \log n$$

Sia $m = \log n$, ovvero $n = 2^m$

$$T\left(2^m\right) = 2T\left(\sqrt{2^m}\right) + \log 2^m \Rightarrow 2T\left(2^{m/2}\right) + m$$

Ora introduciamo la funzione $S\left(m\right) = T\left(2^m\right)$

$$S\left(m\right) = 2S\left(\frac{m}{2}\right) + m$$

che è $S\left(m\right) \in \Theta\left(m \log m\right)$

Ma allora $T\left(n\right) \in \Theta\left(\log n \log \log n\right)$!

Come trovare l'espressione giusta?

5. rafforzare l'ipotesi induttiva

$$T(n) = \begin{cases} 1 & \text{per } n = 1 \\ 2T\left(\frac{n}{2}\right) + 1 & \text{per } n > 1 \end{cases}$$

Viene spontaneo ipotizzare $T(n) \in O(n)$, cioè

$$\exists n_0 \in \mathbb{N}, c > 0 : T(n) \leq cn \quad \forall n \geq n_0$$

Se poniamo per semplicità $n_0 = \bar{n}_0 = 1$, la tesi del passo base è

Tesi 1: $T(1) = 1 \leq c \cdot 1$

e per dimostrarla basta scegliere $c \geq 1$

Come trovare l'espressione giusta?

Il passo induttivo ipotizza $T(i) \leq ci \quad \forall i : n_0 \leq i \leq n-1$ e ha

Tesi 2: $T(n) \leq cn$

$$T(n) = 2\,T\left(\frac{n}{2}\right) + 1 \Rightarrow$$

$$\Rightarrow T(n) \leq 2\,c\frac{n}{2} + 1 = cn + 1 \not\leq cn$$

Questa tesi non si può dimostrare per nessun valore di c.

Allora ne dimostriamo una più forte!

Se il primo gradino è più alto si sale di più con passi più piccoli:

se il caso base è più forte, si dimostra una tesi più forte

con passi induttivi più deboli

Rafforzare l'ipotesi induttiva

Dimostriamo che

$$\exists n_0 \in \mathbb{N}, c > 0, c' > 0 : T(n) \leq cn - c' \quad \forall n \geq n_0$$

Tesi 1: $T(1) = 1 \leq c \cdot 1 - c'$, dimostrata per $c \geq c' + 1$

Il passo induttivo ipotizza $T(i) \leq ci - c'$ per $i = n_0, \ldots, n-1$
Tesi 2: $T(n) \leq cn - c'$

Con i consueti passaggi, $n > n_0 = 1 \Rightarrow n/2 \leq n - 1 \Rightarrow$

$$\Rightarrow T(n) = 2T\left(\frac{n}{2}\right) + 1 \leq 2\left(c\frac{n}{2} - c'\right) + 1 = cn - 2c' + 1$$

Basta porre $c' = 1$ e $c = 2$ e si ottiene $T(n) \leq cn - c'$

Il metodo iterativo

$$T(n) = \begin{cases} c & n = 1 \\ \Phi\left(T(n_1), \ldots, T(n_k)\right) & n > n_i \ (i = 1, \ldots, k) \end{cases}$$

Per "indovinare" la soluzione di un'equazione ricorrente si può

1. ricavare le $T(n_i)$ dal membro di sinistra e sostituirle nel membro di destra sino a raggiungere il caso base
2. sostituire l'espressione del caso base
3. valutare l'espressione risultante

Esempio: Fattoriale

Fattoriale(n)

If $n \leq 1$

then **Return** 1; { caso base }

else $a = n - 1$ { Divide }

b := Fattoriale(a); { Impera }

Return $n \cdot b$; { Combina }

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ \Theta(1) + T(n-1) + \Theta(1) & n > 1 \end{cases}$$

Esempio per il metodo iterativo

$$T(n) = \begin{cases} a & n = 1 \\ T(n-1) + b & n > 1 \end{cases}$$

1. ricavare le $T(n_i)$ dal membro di sinistra e sostituirle nel membro di destra sino a raggiungere il caso base

$$\begin{aligned}
T(n) &= T(n-1) + b &&= T(n-1) + b = \\
&= \overbrace{T(n-2) + b} + b &&= T(n-2) + 2b = \\
&= \overbrace{T(n-3) + b} + 2b &&= T(n-3) + 3b = \\
&= \dots \\
&= \overbrace{T(n-(n-1)) + b} + (n-2)\,b &&= T(1) + (n-1)\,b
\end{aligned}$$

Esempio per il metodo iterativo

2. sostituire l'espressione del caso base

$$T(n) = T(1) + (n-1)\,b = a + (n-1)\,b$$

3. valutare l'espressione risultante

$$T(n) = bn + a - b \in O(n)$$

Un altro esempio iterativo

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ n + 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) & n > 1 \end{cases}$$

$$T(n) = n + 3 \quad T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) =$$

$$= n + 3 \overbrace{\left[\left\lfloor \frac{n}{4} \right\rfloor + 3T\left(\left\lfloor \frac{\left\lfloor \frac{n}{4} \right\rfloor}{4} \right\rfloor\right)\right]}$$

Poiché $\left\lfloor \frac{\left\lfloor \frac{n}{a} \right\rfloor}{b} \right\rfloor = \left\lfloor \frac{n}{ab} \right\rfloor$, è $T(n) = n + 3\left\lfloor \frac{n}{4} \right\rfloor + 9T\left(\left\lfloor \frac{n}{16} \right\rfloor\right)$

Un altro esempio iterativo

$$
\begin{aligned}
T(n) &= n + 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) = \\
&= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 9T\left(\left\lfloor \frac{n}{16} \right\rfloor\right) = \\
&= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 9\left\lfloor \frac{n}{16} \right\rfloor + 27T\left(\left\lfloor \frac{n}{64} \right\rfloor\right) = \\
&= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 3^2\left\lfloor \frac{n}{4^2} \right\rfloor + 3^3T\left(\left\lfloor \frac{n}{4^3} \right\rfloor\right) = \\
&= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 3^2\left\lfloor \frac{n}{4^2} \right\rfloor + \ldots + 3^{h-1}\left\lfloor \frac{n}{4^{h-1}} \right\rfloor + 3^hT\left(\left\lfloor \frac{n}{4^h} \right\rfloor\right)
\end{aligned}
$$

Ci si ferma quando $\left\lfloor \frac{n}{4^h} \right\rfloor \leq 1$, cioè $h = \lfloor \log_4 n \rfloor$

Un altro esempio iterativo

$$T(n) = \sum_{i=0}^{\log_4 n - 1} 3^i \left\lfloor \frac{n}{4^i} \right\rfloor + 3^{\log_4 n} T\left(\left\lfloor \frac{n}{4^{\log_4 n}} \right\rfloor \right) =$$

$$\leq \sum_{i=0}^{\log_4 n - 1} \frac{3^i}{4^i} n + 3^{\log_4 n} T(1) =$$

$$\leq n \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{4} \right)^i + 3^{\log_4 n} \Theta(1) =$$

Maggiorando la somma con una serie e ricordando $a^{\log_b n} = n^{\log_b a}$

$$T(n) < n \sum_{i=0}^{+\infty} \frac{3^i}{4^i} + \Theta\left(n^{\log_4 3} \right) = 4n + \Theta\left(n^{\log_4 3} \right)$$

$T(n) \in \Theta(n)$, perché $4n$ cresce più in fretta di $n^{\log_4 3}$

Aspetti chiave

I punti fondamentali del metodo iterativo sono

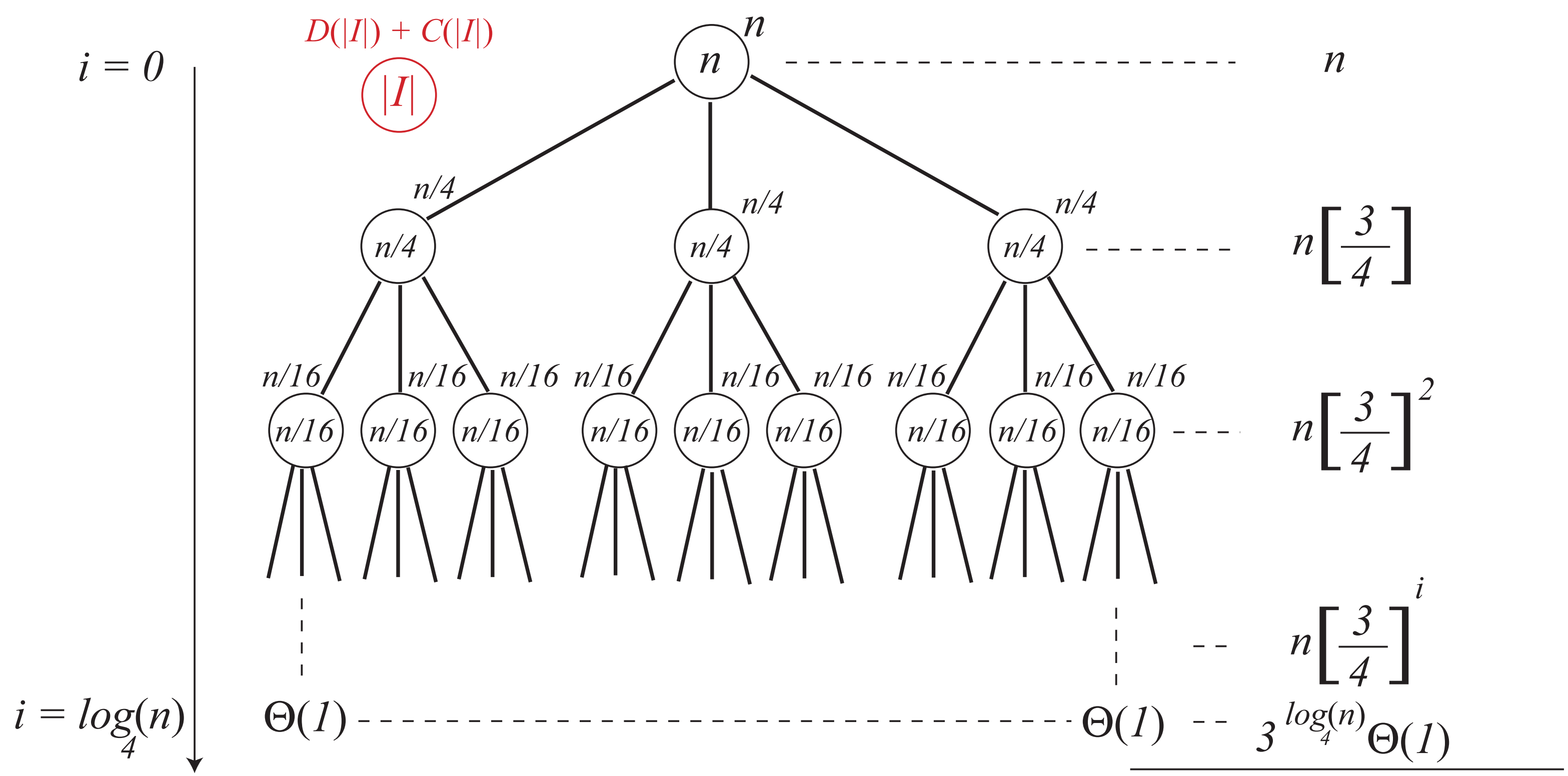
- quante volte bisogna iterare la ricorrenza prima di giungere al caso base (profondità h della ricorsione)
- quanto vale la somma dei termini prodotti a ciascun livello dell'iterazione

Gli alberi di ricorsione semplificano fortemente quest'analisi

- sforzo nei nodi interni: sommatoria dello sforzo computazionale per ogni livello da $i = 0$ a $i = h - 1$
- sforzo nelle foglie: numero delle foglie moltiplicato per lo sforzo computazionale nel caso base

Alberi di ricorsione (esempio 1)

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ n + 3T\left(\frac{n}{4}\right) & n > 1 \end{cases}$$

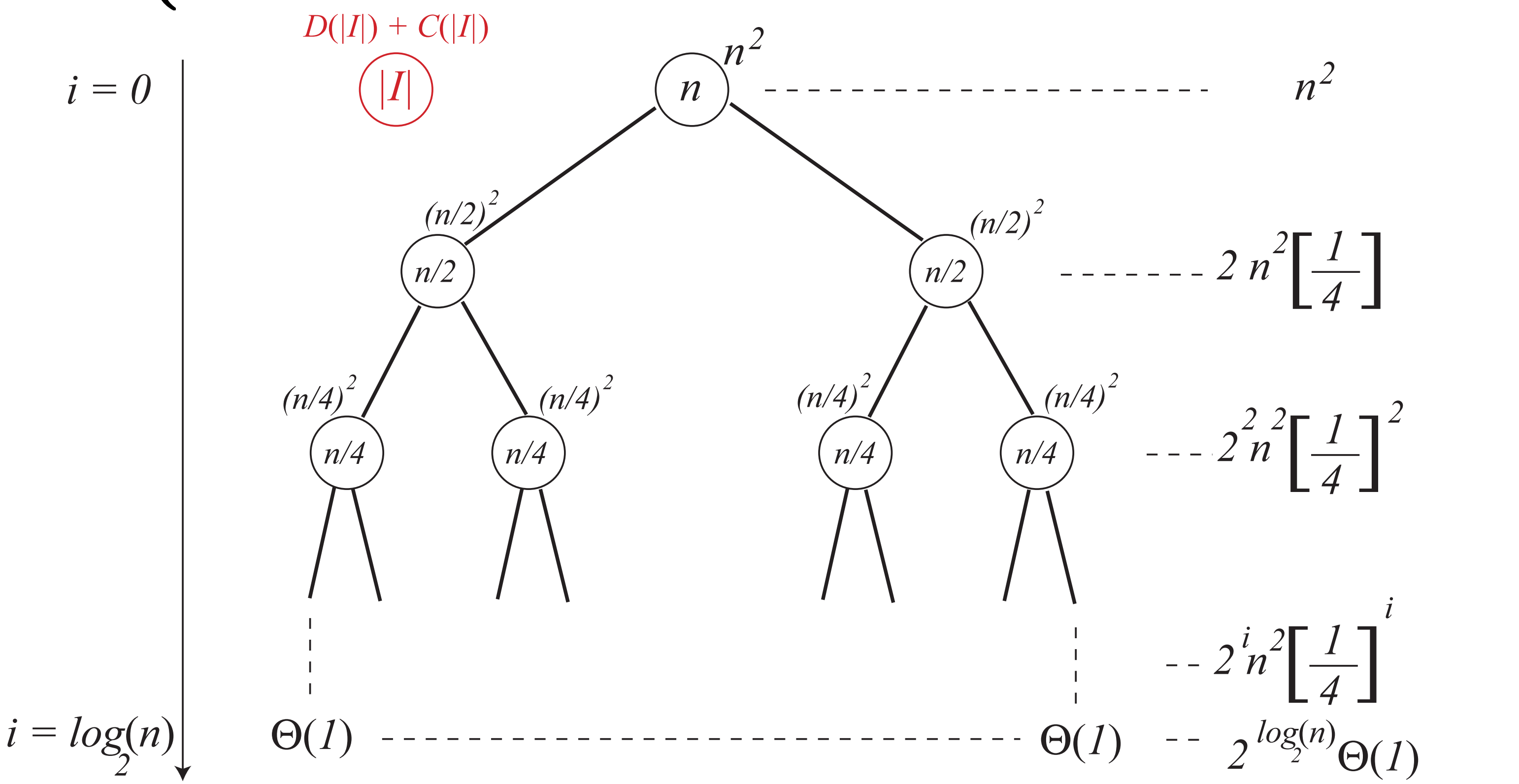


$$\sum_{i=0}^{\log_4(n)-1} n\left[\frac{3}{4}\right]^i + 3^{\log_4(n)}\Theta(1)$$

Profondità h: $\dfrac{n}{4^h} \leq 1 \Rightarrow h \geq \log_4 n$

Alberi di ricorsione (esempio 2)

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ n^2 + 2T\left(\frac{n}{2}\right) & n > 1 \end{cases}$$



$$\sum_{i=0}^{\log_2(n)-1} n^2 \left[\frac{1}{2}\right]^i + 2^{\log_2(n)}\Theta(1)$$

Profondità h: $\frac{n}{2^h} \leq 1 \Rightarrow h \geq \log_2 n$

Alberi di ricorsione (esempio 2)

$$T(n) = \sum_{i=0}^{\log_2 n-1} n^2 \left(\frac{1}{2}\right)^i + 2^{\log_2 n}\Theta(1)$$

Ricordando che

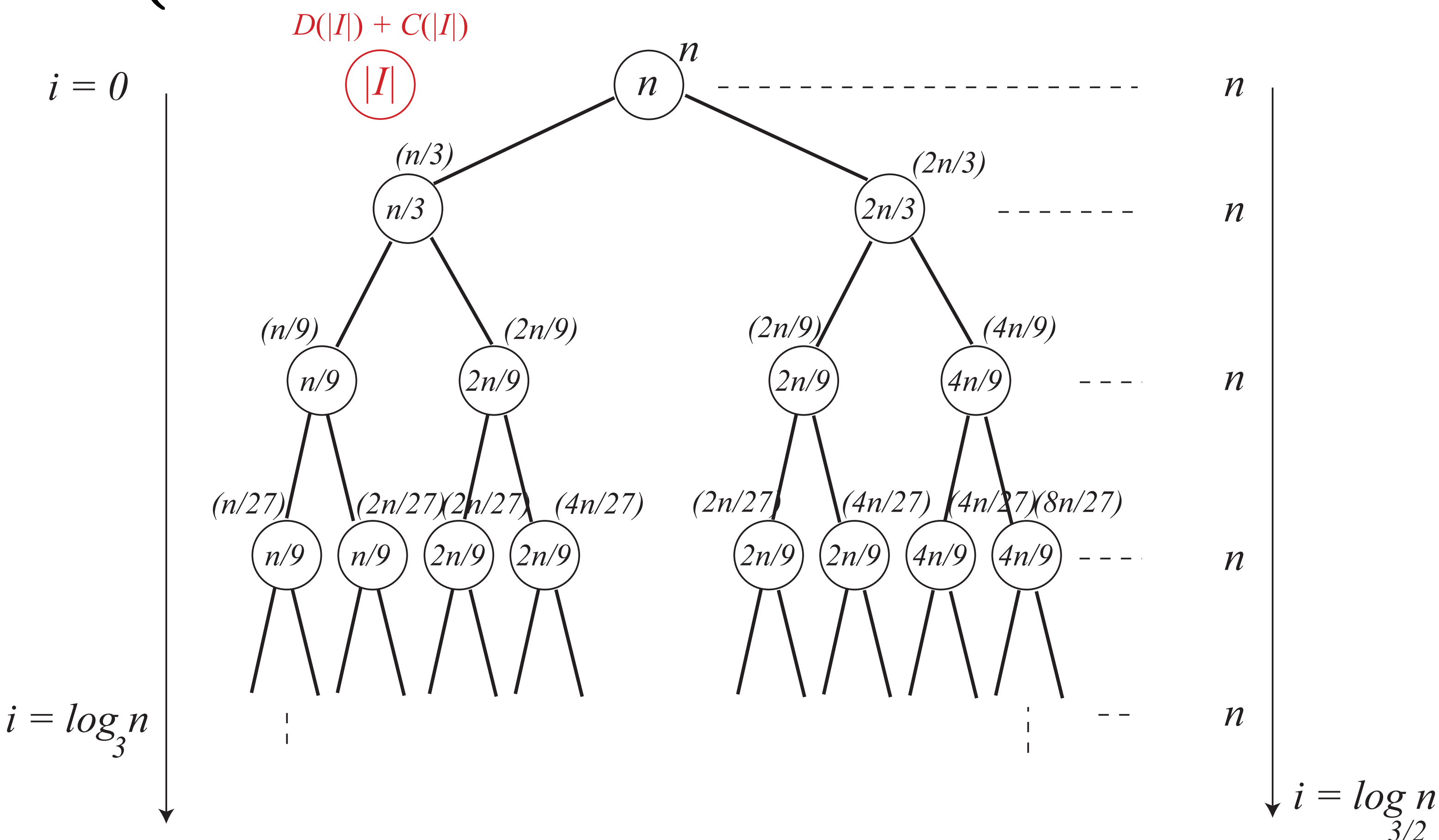
$$\sum_{i=0}^{k} x^i = \frac{x^{k+1}-1}{x-1} \quad \text{e} \quad 2^{\log_2 n} = n$$

si ricava

$$T(n) = n^2 \frac{\left(\frac{1}{2}\right)^{\log_2 n} - 1}{\left(\frac{1}{2}\right) - 1} + \Theta(n) = 2n^2 \frac{n-1}{n} + \Theta(n) = \Theta\left(n^2\right)$$

Alberi di ricorsione (esempio 3)

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ n + T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) & n > 1 \end{cases}$$



Alberi di ricorsione (esempio 3)

Questa volta l'albero non è bilanciato

- sul lato sinistro: profondità h: $\frac{n}{3^h} \leq 1 \Rightarrow h \geq \log_3 n$
- sul lato destro: profondità h: $\left(\frac{2}{3}\right)^h n \leq 1 \Rightarrow h \geq \log_{3/2} n$

Quindi

- fermandosi al livello $h_1 = \log_3 n$, si ha una stima per difetto
- arrivando al livello $h_2 = \log_{3/2} n$, si ha una stima per eccesso

Essendo n lo sforzo computazionale ad ogni livello completo

$$(h_1 + 1)\, n \leq T(n) \leq (h_2 + 1)\, n \Rightarrow$$

$$\Rightarrow (\log_3 n + 1)\, n \leq T(n) \leq \Big(\log_{3/2} n + 1\Big)\, n \Rightarrow$$

$$\Rightarrow \frac{\log_2 n}{\log_2 3} n \leq T(n) \leq \frac{\log_2 n}{\log_2 3/2} n + n,$$ da cui $T(n) \in \Theta(n \log n)$

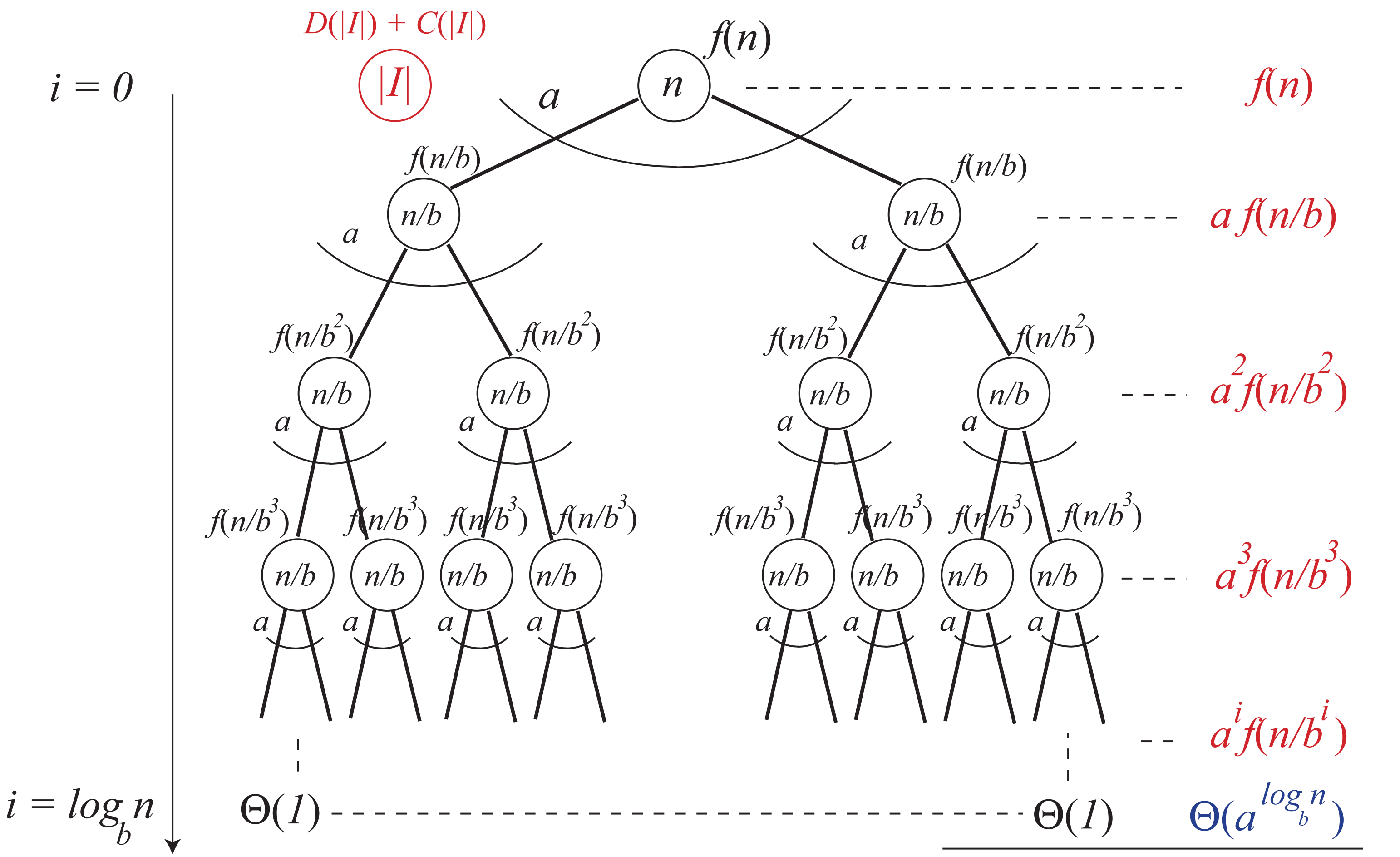
Il teorema principale

$$T(n) = \begin{cases} k & n = \bar{n} \\ aT\left(\frac{n}{b}\right) + f(n) & n > \bar{n} \end{cases}$$

con $k \in \mathbb{R}$, $a \geq 1$, $b > 1$ e $f(n)$ asintoticamente positiva

1. Se $\exists \epsilon > 0 : f(n) \in O\left(n^{\log_b a - \epsilon}\right)$, allora $T(n) \in \Theta\left(n^{\log_b a}\right)$
2. Se $f(n) \in \Theta\left(n^{\log_b a}\right)$, allora $T(n) \in \Theta\left(n^{\log_b a} \log n\right) = \Theta(f(n) \log n)$
3. Se $\begin{cases} \exists \epsilon > 0 : f(n) \in \Omega\left(n^{\log_b a + \epsilon}\right) \\ \exists c \in (0;1), n_0 \in \mathbb{N} : af(n/b) \leq cf(n), \forall n \geq n_0 \end{cases}$
   allora $T(n) \in \Theta(f(n))$

Dimostrazione (traccia)



$$\sum_{i=0}^{\log_b(n)-1} a^i f(n/b^i) + \Theta(n^{\log_b a})$$

Profondità h: $\dfrac{n}{b^h} \leq 1 \Rightarrow h \geq \log_b n$

Motivazione intuitiva

L'idea è confrontare

- il tempo $f(n)$ impiegato a scomporre il problema e ricomporre la soluzione alla radice
- il tempo $n^{\log_b a}$ impiegato a risolvere tutti i casi base

1. prevale $n^{\log_b a} \Rightarrow T(n) = \Theta\left(n^{\log_b a}\right)$
2. si equivalgono (anche con i livelli intermedi) $\Rightarrow T(n) \in \Theta(f(n) \log n)$
3. prevale $f(n)$ (e ogni livello prevale sul seguente) $\Rightarrow T(n) = \Theta(f(n))$

Prevale = c'è un fattore polinomiale n^{ϵ} di differenza

Osservazioni

2. La condizione

$$f(n) \in \Theta\left(n^{\log_b a}\right)$$

indica che ogni livello pesa come gli altri, ad es. la radice
$\Rightarrow$ si moltiplica lo sforzo alla radice per il numero dei livelli

3. La condizione

$$\exists c \in (0;1), n_0 \in \mathbb{N} : af(n/b) \leq cf(n), \forall n \geq n_0$$

indica che ciascun livello domina il successivo
$\Rightarrow$ ci si limita alla radice

Esempi (1)

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T\left(\frac{n}{2}\right) + n & n > 1 \end{cases}$$

- $f(n) = n$ asintoticamente positiva
- $a = 2 \geq 1$ e $b = 2 > 1 \Rightarrow n^{\log_b a} = n^{\log_2 2} = n$
- $f(n) \in \Theta\left(n^{\log_b a}\right) \Rightarrow$ caso 2

$$\Rightarrow T(n) \in \Theta(n \log n)$$

Esempi (2)

$$T(n) = \begin{cases} 1 & n = 1 \\ 9T\left(\frac{n}{3}\right) + n & n > 1 \end{cases}$$

- $f(n) = n$ asintoticamente positiva
- $a = 9 \geq 1$ e $b = 3 > 1 \Rightarrow n^{\log_b a} = n^{\log_3 9} = n^2$
- $f(n) \in O\left(n^{\log_b a - \epsilon}\right)$, dato che $n^{\log_3 9 - \epsilon} = n^{2-\epsilon}$ e $n \in O\left(n^{2-\epsilon}\right)$ per ogni $\epsilon \leq 1$; quindi siamo nel caso 1

$$\Rightarrow T(n) \in \Theta\left(n^2\right)$$

Esempi (3)

$$T(n) = \begin{cases} 1 & n = 1 \\ T\left(\frac{2n}{3}\right) + 1 & n > 1 \end{cases}$$

- $f(n) = 1$ asintoticamente positiva
- $a = 1 \geq 1$ e $b = 3/2 > 1 \Rightarrow n^{\log_b a} = n^{\log_{3/2} 1} = 1$
- $f(n) \in \Theta\left(n^{\log_b a}\right) \Rightarrow$ caso 2

$$\Rightarrow T(n) \in \Theta(\log n)$$

Esempi (4)

$$T(n) = 3T\left(\frac{n}{4}\right) + n \log n$$

- $f(n) = n \log n$ asintoticamente positiva
- $a = 3 \geq 1$ e $b = 4 > 1 \Rightarrow n^{\log_b a} = n^{\log_4 3}$
- $f(n) \in \Omega\left(n^{\log_b a} + \epsilon\right)$, dato che $n \log n \in \Omega\left(n^{\log_4 3+\epsilon}\right)$ per ogni $\epsilon \leq 1 - \log_4 3$: caso 3?
- Condizioni di regolarità: $af(n/b) \leq cf(n)$ diventa

$$3\frac{n}{4}\log\frac{n}{4} \leq cn\log n \Leftrightarrow \frac{3}{4}n\log n - \frac{3}{4}n\log 4 \leq cn\log n$$

Basta porre $c = 3/4$ per verificarla $\Rightarrow$ caso 3

$$\Rightarrow T(n) \in \Theta(n \log n)$$

Esempi (5)

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T\left(\frac{n}{4}\right) + n & n > 1 \end{cases}$$

- $f(n) = n$ asintoticamente positiva
- $a = 3 \geq 1$ e $b = 4 > 1 \Rightarrow n^{\log_b a} = n^{\log_4 3}$
- $f(n) \in \Omega\left(n^{\log_b a} + \epsilon\right)$, dato che $n \in \Omega\left(n^{\log_4 3+\epsilon}\right)$ per ogni $\epsilon \leq 1 - \log_4 3$: caso 3?
- Condizioni di regolarità: $af(n/b) \leq cf(n)$ diventa

$$3\frac{n}{4} \leq cn$$

Basta porre $c = 3/4$ per verificarla $\Rightarrow T(n) \in \Theta(n)$

Esempi (6)

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T\left(\frac{n}{2}\right) + n\log n & n > 1 \end{cases}$$

- $f(n) = n\log n$ asintoticamente positiva
- $a = 2 \geq 1$ e $b = 2 > 1 \Rightarrow n^{\log_b a} = n^{\log_2 2} = n$
- $f(n) \in \Omega\left(n^{\log_b a}\right)$, ma se $\epsilon > 0$, $f(n) \notin \Omega\left(n^{\log_b a+\epsilon}\right)$; infatti

$$\frac{f(n)}{n^{\log_b a}} = \log n$$

che non domina n^{ϵ} per alcun valore di ϵ

Il teorema non vale!

Estensione del teorema principale

$$T(n) = \begin{cases} k & n = \bar{n} \\ aT\left(\frac{n}{b}\right) + f(n) & n > \bar{n} \end{cases}$$

con $k \in \mathbb{R}$, $a \geq 1$, $b > 1$ e $f(n)$ asintoticamente positiva

Se $\exists h \geq 0 : f(n) \in \Theta\left(n^{\log_b a} \log^h n\right)$, allora
$T(n) \in \Theta\left(n^{\log_b a} \log^{h+1} n\right) = \Theta(f(n) \log n)$

$$T(n) = 2T\left(\frac{n}{2}\right) + n \log n$$

- $f(n) = n \log n$ asintoticamente positiva
- $a = 2 \geq 1$ e $b = 2 > 1 \Rightarrow n^{\log_b a} = n^{\log_2 2} = n$
- $f(n) \in \Theta\left(n^{\log_b a} \log^h n\right)$ per $h = 1$ (estensione)
  $\Rightarrow T(n) \in \Theta\left(n \log^2 n\right)$