5) Equazioni di ricorrenza

Sommario: In questo capitolo vengono forniti alcuni metodi per risolvere le equazioni di ricorrenza, indispensabili per il calcolo del tempo computazionale degli algoritmi ricorsivi.

Valutare il costo computazionale di un algoritmo ricorsivo è, in genere, più laborioso che nel caso degli algoritmi iterativi.

Infatti, la natura ricorsiva della soluzione algoritmica dà luogo a una funzione di costo che, essendo strettamente legata alla struttura dell'algoritmo, è anch'essa ricorsiva. Trovare la funzione di costo ricorsiva è piuttosto immediato. Essa però deve essere riformulata in modo che non sia più ricorsiva, altrimenti il costo asintotico non può essere quantificato, e questa è la parte meno semplice. La riformulazione della funzione di costo ricorsiva in una equivalente ma non ricorsiva si affronta impostando una ***equazione di ricorrenza***, costituita dalla formulazione ricorsiva e dal caso base.

Iniziamo da un esempio, quello della ricerca sequenziale ricorsiva (par. 4.2.2). Il corpo della funzione, chiamato su un vettore di dimensione n:

- effettua un test;
- se tale test non è soddisfatto effettua una chiamata ricorsiva su un vettore di $(n - 1)$ elementi.

Dunque, nel corpo della funzione abbiamo un numero costante di operazioni elementari e una chiamata ricorsiva.

Indicando con $T(n)$ il costo computazionale dell'algoritmo ricorsivo, possiamo esprimerlo mediante questa equazione di ricorrenza:

- $T(n) = T(n - 1) + \Theta(1)$
- $T(1) = \Theta(1)$

In generale, la parte generica dell'equazione di ricorrenza che definisce $T(n)$ deve essere sempre costituita dalla somma di almeno due addendi, di cui

almeno uno contiene la parte ricorsiva (nell'esempio sopra $T(n-1)$) mentre uno rappresenta il costo computazionale di tutto ciò che viene eseguito al di fuori della chiamata ricorsiva. Si tenga presente che, anche se questa parte dovesse essere un solo confronto, il suo costo non può essere ignorato, altrimenti si otterrebbe che la funzione ha un costo computazionale indipendente dalla dimensione del suo input, e questa è data da quanto illustrato nel caso base. Non possiamo dire che ciò sia impossibile, ma è molto improbabile.

Esistono alcuni metodi utili per risolvere le equazioni di ricorrenza, che saranno illustrati nei prossimi paragrafi:

- metodo di sostituzione,
- metodo iterativo,
- metodo dell'albero,
- metodo principale.

5.1 Metodo di sostituzione

L'idea alla base di questo metodo è la seguente:

- si ipotizza una soluzione per l'equazione di ricorrenza data;
- si verifica se essa "funziona" procedendo per induzione.

La difficoltà di questo metodo risiede nel fatto che si deve trovare la funzione più vicina alla vera soluzione, perché tutte le funzioni più grandi (se stiamo cercando O) o più piccole (se stiamo cercando Ω) funzionano. In effetti questo metodo serve soprattutto nelle dimostrazioni mentre si sconsiglia di utilizzarlo nella pratica.

Applichiamo questo metodo all'equazione di ricorrenza dell'algoritmo ricorsivo di ricerca sequenziale:

- $T(n) = T(n-1) + \Theta(1)$
- $T(1) = \Theta(1)$

Ipotizziamo la soluzione $T(n)=cn$ per una costante opportuna c. Sostituendo, otteniamo: $T(n)=cn=c(n-1)+\Theta(1)$, cioè $c=\Theta(1)$; tuttavia, sostituendo nel caso base abbiamo anche $T(1)=c=\Theta(1)$, ma non è affatto detto che le due costanti nascoste nella notazione asintotica siano le stesse. Per questa ragione, è

innanzi tutto necessario eliminare la notazione asintotica dall'equazione di ricorrenza, così che le costanti non rimangano "nascoste" nella notazione, conducendo a risultati errati: infatti la verifica della soluzione va fatta in termini esatti e non asintotici.

Quindi, l'equazione di ricorrenza può essere trasformata, senza cambiarne il significato, nella seguente:

- $T(n) = T(n - 1) + c$
- $T(1) = d$

per due costanti c e d positive fissate.

Ipotizziamo ora la soluzione $T(n) = O(n)$, *ossia* $T(n) \leq kn$ dove k è una costante che va ancora determinata. Si osservi che non si può sperare in una soluzione esatta, ma possiamo solo maggiorare o minorare.

Procediamo quindi per induzione.

Caso base. Sostituiamo dapprima la soluzione ipotizzata nel caso base. Si ottiene: $T(1) \leq k$; poiché sapevamo che $T(1) = d$, la disuguaglianza è soddisfatta se e solo se $k \geq d$.

Passo induttivo. Se assumiamo che la soluzione $T(r) \leq kr$ sia vera per ogni $r<n$, sostituiamo la soluzione nella formulazione ricorsiva dell'equazione di ricorrenza e cerchiamo di dimostrare che essa è vera per n:

$$T(n) \leq k(n-1) + c = kn - k + c \leq kn$$

L'ultima disuguaglianza è vera se e solo se $k \geq c$.

La nostra soluzione è dunque corretta per tutti i valori di k tali che $k \geq c$ e $k \geq d$. Poiché un tale valore di k esiste sempre, una volta fissati c e d, possiamo concludere che $T(n)$ è un $O(n)$.

Si noti che, se avessimo ipotizzato $T(n) \leq kn^2$, con operazioni simili avremmo trovato che anche questa soluzione è corretta per tutti i valori di k tali che $k \geq c$ e $k \geq d$. Ma, come già detto, a noi interessa stimare $T(n)$ asintoticamente tramite la funzione più piccola, e questo è spesso un obiettivo difficile.

Per rendere il nostro risultato stretto, ipotizziamo ora la soluzione $T(n) = \Omega(n)$, *ossia* $T(n) \geq hn$ dove h è una costante che va ancora determinata.

In modo assolutamente analogo al caso O, sostituiamo dapprima la soluzione ipotizzata nel caso base ottenendo $h \leq d$, e poi nella formulazione ricorsiva dell'equazione di ricorrenza ottenendo invece:

$$T(n) \geq h(n-1) + c = hn - h + c \geq hn$$

vera se e solo se $h \leq c$.

Deduciamo dunque che $T(n)$ è un $\Omega(n)$ poiché è possibile trovare un valore h ($h \leq c$, $h \leq d$) *per il quale si ha* $T(n) \geq hn$.

Dalle due soluzioni: $T(n) = O(n)$ e $T(n) = \Omega(n)$, si ottiene ovviamente che $T(n) = \Theta(n)$.

Concludiamo osservando che quanto avvenuto nel precedente esempio non è casuale: quando si usa il metodo di sostituzione, non si può mai ottenere immediatamente una soluzione che sia Θ di una qualche funzione, ma si può solo sperare di ottenere una soluzione che sia O (usando il $\leq$) oppure che sia Ω (usando il $\geq$).

Esempio 5.1

Sia data la seguente equazione di ricorrenza:

- $T(n)=2T(n/2) + \Theta(1)$
- $T(1)= \Theta(1)$

Dobbiamo innnazitutto eliminare la notazione asintotica, quindi l'equazione diventa:

$T(n)=2T(n/2)+c$ per qualche costante c>0

$T(1)=d$ per qualche costante d>0

Proviamo a dimostrare per induzione che la soluzione è $T(n) \leq kn$, dove k è una costante da determinare.

Passo base. $T(1) \leq k$, ma $T(1) = d$ se e solo se $k \geq d$.

Passo induttivo. Sostituendo nell'equazione generica otteniamo:

$$T(n) \leq 2\,(k\,n/2) + c = kn + c$$

---

che non è mai limitata superiormente da *kn.* Questo tuttavia non vuole sempre dire che la soluzione non sia esatta, ma che la dobbiamo scegliere in modo più oculato.

Prendiamo, ad esempio, la soluzione $T(n) \leq kn-h$, dove k e h sono costanti da determinare.

Sostituendo, otteniamo:

$$T(n) \leq 2(k\, n/2-h) + c = kn - 2h + c = kn-h + (c-h) \leq kn-h \text{ sse } c-h \leq 0 \text{ cioè } c \leq h$$

Sostituendo nel caso base, otteniamo:

$$d=T(1) \leq k-h \text{ vera sse } d \leq k-h, \text{ cioè } d+h \leq k$$

e quindi che il passo base è verificato. Concludiamo quindi che $T(n)=O(n)$.

In modo analogo, possiamo dimostrare che $T(n)=\Omega(n)$.

5.2 Metodo iterativo

L'idea alla base di questo metodo è di sviluppare l'equazione di ricorrenza ed esprimerla come somma di termini dipendenti da n e dal caso base.

Rispetto al metodo di sostituzione questo metodo richiede una maggiore quantità di calcoli algebrici ma è più meccanico.

Applichiamo questo metodo alla stessa equazione di ricorrenza vista prima, quella dell'algoritmo ricorsivo di ricerca sequenziale:

- $T(n) = T(n-1) + \Theta(1)$
- $T(1) = \Theta(1)$

In questo semplice caso, "srotolando" ripetutamente la parte destra della prima equazione otterremo:

$$T(n) = T(n-1) + \Theta(1), \text{ ma } T(n-1)=T(n-2) + \Theta(1) \text{ quindi}$$

$$T(n) = T(n-2) + \Theta(1) + \Theta(1), \text{ ma } T(n-2)=T(n-3) + \Theta(1) \text{ quindi}$$

$$T(n) = T(n-3) + \Theta(1) + \Theta(1) + \Theta(1), \text{ ma } T(n-3)=T(n-4) + \Theta(1) \text{ quindi}$$

$$T(n) = T(n-4) + \Theta(1) + \Theta(1) + \Theta(1) + \Theta(1), \text{ ecc.}$$

fino a ottenere:

$$T(n) = n\,\Theta(1) = \Theta(n).$$

Come si vede, otteniamo una soluzione identica a quella trovata col metodo di sostituzione.

Applichiamo ora il metodo iterativo all'equazione di ricorrenza che descrive la il costo computazionale dell'algoritmo di ricerca binaria ricorsiva (par. 4.2.3).

L'equazione di ricorrenza è la seguente:

- $T(n) = T(n/2) + \Theta(1)$
- $T(1) = T(0) = \Theta(1)$

in quanto, nel corpo della funzione, troviamo un numero costante di operazioni elementari ed una chiamata ricorsiva che opera su un problema di dimensione dimezzata.

Sviluppiamo l'equazione:

$$T(n) = T(\tfrac{n}{2}) + \Theta(1) = T(\tfrac{n}{2^2}) + \Theta(1) + \Theta(1) = T(\tfrac{n}{2^3}) + \Theta(1) + \Theta(1) + \Theta(1) =$$

$$= \ldots = T(\tfrac{n}{2^k}) + k\,\Theta(1)$$

Ora, quando $k = \log n$ si ha $\frac{n}{2^k} = 1$, quindi per tale valore di k ci fermiamo ed otteniamo:

$$T(n) = \Theta(1) + \log n\,\Theta(1) = \Theta(\log n).$$

Vediamo infine un altro esempio, l'algoritmo ricorsivo per il calcolo dei numeri di Fibonacci (par. 4.2.4). Nel corpo della funzione abbiamo un numero costante di operazioni e due chiamate ricorsive.

L'equazione di ricorrenza è perciò:

- $T(n) = T(n-1) + T(n-2) + \Theta(1)$
- $T(1) = T(0) = \Theta(1)$

E' facile vedere che non si riesce a risolvere questa equazione di ricorrenza tramite il metodo iterativo, in quanto il numero di addendi cresce esponenzialmente giungendo presto ad una quantità non gestibile. Possiamo però fare le seguenti considerazioni:

- $T(n) \leq 2T(n-1) + \Theta(1)$: questa disuguaglianza ci permetterà di derivare un limite superiore O;
- $T(n) \geq 2T(n-2) + \Theta(1)$: questa disuguaglianza ci permetterà di derivare un limite inferiore Ω.

Limite superiore

Applichiamo il metodo di iterazione alla prima disuguaglianza. Otteniamo:

$$T(n) \leq 2T(n-1) + \Theta(1) \leq$$

$$\leq 2^2T(n-2) + 2^1\Theta(1) + \Theta(1) \leq$$

$$\leq 2^3T(n-3) + 2^2\Theta(1) + 2^1\Theta(1) + \Theta(1) \leq$$

$$\dots$$

$$\leq 2^kT(n-k) + \sum_{i=0}^{k-1} 2^i\,\Theta(1)$$

Il procedimento si ferma quando $n - k = 1$, ossia $k = n - 1$ per cui otteniamo:

$$T(n) \leq 2^{n-1}\,\Theta(1) + \sum_{i=0}^{n-2} 2^i\,\Theta(1) = 2^{n-1}\,\Theta(1) + (2^{n-1} - 1)\Theta(1) = (2^n - 1)\Theta(1)$$

E dunque troviamo che:

$$T(n) = O(2^n).$$

Limite inferiore

Applichiamo il metodo di sostituzione alla seconda disuguaglianza. Otteniamo:

$$T(n) \geq 2T(n-2) + \Theta(1) \geq$$

$$\geq 2^2T(n-4) + 2^1\Theta(1) + \Theta(1) \geq$$

$$\geq 2^3T(n-6) + 2^2\Theta(1) + 2^1\Theta(1) + \Theta(1) \geq$$

$$\dots$$

$$\geq 2^kT(n-2k) + \sum_{i=0}^{k-1} 2^i\,\Theta(1)$$

Il procedimento si ferma quando $n - 2k = 0$ (se n è pari; per n dispari il procedimento termina quando $n - 2k = 1$ e differisce per alcuni dettagli, ma il comportamento asintotico non cambia), ossia $k = n/2$ per cui otteniamo:

$$T(n) \geq 2^{n/2}\,\Theta(1) + \sum_{i=0}^{\frac{n}{2}-1} 2^i\,\Theta(1) = 2^{n/2}\,\Theta(1) + (2^{n/2} - 1)\Theta(1) = (2*2^{n/2} - 1)\Theta(1)$$

E dunque troviamo che:

$$T(n) = \Omega(2^{n/2}).$$

Osserviamo che, anche se non siamo in grado di trovare una funzione asintotica precisa (Θ), possiamo comunque concludere che il calcolo dei numeri di Fibonacci con una tecnica ricorsiva richiede un tempo esponenziale in n, visto che $k_1 2^{n/2} \leq T(n) \leq k_2 2^n$ per opportune costanti k_1 e k_2.

5.3 Metodo dell'albero

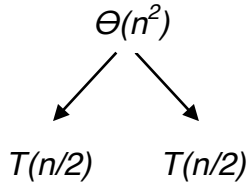
La costruzione dell'***albero di ricorrenza*** di un algoritmo ricorsivo è una tecnica per rappresentare graficamente lo sviluppo del costo computazionale dell'algoritmo, in modo da poterlo valutare con una certa facilità.

Ogni nodo dell'albero è relativo alla soluzione di un problema di una certa dimensione. Esso riporta il costo della ricombinazione delle soluzioni parziali, ed ha tanti figli quanti sono i sottoproblemi in cui il problema relativo al nodo stesso è ricorsivamente scomposto.

Ad esempio, per la seguente equazione di ricorrenza:

- $T(n) = 2T(n/2) + \Theta(n^2)$
- $T(1) = \Theta(1)$

si deriva l'albero di ricorrenza nel seguente modo. Si costruisce la radice coi suoi figli:



in modo tale che sulla radice ci sia il contributo relativo a tutta la parte dell'algoritmo che non sia ricorsiva (in questo caso $\Theta(n^2)$), e si aggiunge un figlio per ogni chiamata ricorsiva effettuata (in questo caso due), tenendo traccia della dimensione del nuovo problema su cui si richiama l'algoritmo (in questo caso $n/2$ su entrambe le chiamate).

Poi si itera il procedimento su ciascuno dei figli fino ad arrivare alle foglie, che corrispondono ai casi base e quindi non hanno figli. Una volta completato l'albero, il costo computazionale è dato dalla somma dei contributi di tutti i livelli (cioè le "righe" in cui sono disposti i nodi) di cui è costituito l'albero. Sebbene l'albero consenta di visualizzare più chiaramente la decomposizione in sottoproblemi, le problematiche dei calcoli algebrici da effettuare sono essenzialmente analoghe a quelle del metodo iterativo.

Ad esempio, per l'equazione di cui sopra:

livello 1 (radice): $\Theta(n^2)$

livello 2 (figli della radice): $2\,\Theta(\left(\frac{n}{2}\right)^2) = \Theta(\frac{n^2}{4} + \frac{n^2}{4}) = \Theta(\frac{n^2}{2})$

livello 3: $4\Theta(\left(\frac{n}{4}\right)^2) = \Theta(\frac{n^2}{16} + \frac{n^2}{16} + \frac{n^2}{16} + \frac{n^2}{16}) = \Theta(\frac{n^2}{4})$

…

livello i: $2^{i-1}\Theta(\left(\frac{n}{2^{i-1}}\right)^2) = \Theta(\frac{n^2}{2^{i-1}})$

dove *i* ha un valore massimo k tale che $\frac{n}{2^{k-1}} = 1$, ossia $k - 1 = \log n$ e quindi $k = \log n + 1$.

Il contributo totale è quindi:

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

nella seconda espressione abbiamo posto, per convenienza, $j = i - 1$.

Si osservi che, quando calcoliamo il contributo di ciascun livello, non possiamo nascondere le costanti nella notazione Θ, in quanto esse non sono in effetti costanti, come si può evincere dal generico livello *i*, in cui il fattore è 2^{i-1}, che cresce fino a divenire *n*, quando *i* cresce fino a divenire $\log n + 1$.

5.4 Metodo del teorema principale

Questo metodo è molto utile nella pratica e fornisce una soluzione meccanica per le equazioni di ricorrenza della forma:

- $T(n) = aT(n/b) + f(n)$
- $T(1) = \Theta(1)$

Dove $a \geq 1$, $b > 1$ sono delle costanti ed $f(n)$ è una **funzione asintoticamente positiva**.

Si deve subito osservare che l'equazione di ricorrenza precedente non è ben definita, dato che n/b potrebbe non essere un valore intero. E' però facile convincersi che, sostituendo agli a termini $T(n/b)$ i termini $T(\lceil n/b \rceil)$ oppure $T(\lfloor n/b \rfloor)$ il risultato asintotico non cambia. Per questa ragione, commettendo un leggero abuso di notazione, ignoriamo il problema assumendo che n sia sempre una potenza di $b > 1$.

5.4.1 Enunciato del teorema principale

Il teorema principale può essere enunciato come segue:

> Dati $a \geq 1$, $b > 1$, una funzione asintoticamente positiva $f(n)$ ed un'equazione di ricorrenza di forma $T(n) = aT(n/b) + f(n)$, $T(1) = \Theta(1)$ vale che:
>
> 1. Se $f(n) = O(n^{\log_b a - \varepsilon})$ per qualche costante $\boldsymbol{\varepsilon} > 0$ allora $T(n) = \Theta(n^{\log_b a})$
> 2. Se $f(n) = \Theta(n^{\log_b a})$ allora $T(n) = \Theta(n^{\log_b a} \log n)$
> 3. Se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ per qualche costante $\boldsymbol{\varepsilon} > 0$ e se $a\, f(n/b) \leq c\, f(n)$ per qualche costante $c < 1$ e per n sufficientemente grande, allora $T(n) = \Theta(f(n))$

Quanto sopra ci dice che in ciascuno dei tre casi vengono confrontati fra loro $f(n)$ e $n^{\log_b a}$.

Per inciso, mentre già sappiamo che $f(n)$ rappresenta il costo di ricombinazione delle soluzioni ai sottoproblemi, osserviamo ora che il valore $\log_b a$ è legato alla relazione che c'è fra il numero dei sottoproblemi in cui si suddivide un problema e la dimensione dei sottoproblemi in rapporto alla dimensione del problema.

Il teorema principale ci dice che "vince" il maggiore fra $f(n)$ e $n^{\log_b a}$, ossia il costo computazionale è governato dal maggiore dei due:

- se (caso 1) il più grande dei due è $n^{\log_b a}$, allora il costo è $\Theta(n^{\log_b a})$;
- se (caso 3) il più grande dei due è $f(n)$, allora il costo è $\Theta(f(n))$;
- se (caso 2) sono uguali, allora si moltiplica $f(n)$ per un fattore logaritmico.

Si noti che "più grande" e "più piccolo" in questo contesto significa ***polinomialmente*** più grande (o più piccolo), data la posizione all'esponente di ***ε***. In altre parole, *f(n)* deve essere asintoticamente più grande (o più piccola) rispetto a $n^{\log_b a}$ di un fattore n^{ε} per qualche $\varepsilon > 0$.

In effetti fra i casi 1 e 2 vi è un intervallo in cui *f(n)* è più piccola di $n^{\log_b a}$, ma non polinomialmente. Analogamente, fra i casi 2 e 3 vi è un intervallo in cui *f(n)* è più grande di $n^{\log_b a}$, ma non polinomialmente.

In tali intervalli (casi 1 e 2) il metodo del teorema principale non può essere usato, come non può essere usato se (caso 3) non vale la condizione *a f(n/b) ≤ c f(n)*.

Esempio 5.2

$T(n) = 9T(n/3) + \Theta(n)$

- $a = 9,\ b = 3$
- $f(n) = \Theta(n)$
- $n^{\log_b a} = n^{\log_3 9} = n^2$

Poiché $f(n) = \Theta(n^{\log_3 9 - \varepsilon})$ con $\varepsilon = 1$, siamo nel caso 1, per cui $T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$.

Esempio 5.3

$T(n) = T(\frac{2}{3}n) + \Theta(1)$

- $a = 1,\ b = 3/2$
- $f(n) = \Theta(1)$
- $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

Poiché $f(n) = \Theta(n^{\log_b a})$ siamo nel caso 2, per cui $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$.

Esempio 5.4

$T(n) = 3T(n/4) + \Theta(n \log n)$

- $a = 3,\ b = 4$
- $f(n) = \Theta(n \log n)$
- $n^{\log_b a} = n^{\log_4 3} \approx n^{0,7}$

Poiché $f(n) = \Omega(n^{\log_4 3+\varepsilon})$ con, ad esempio, $\varepsilon = 0{,}2$, siamo nel caso 3 se possiamo dimostrare che $3\frac{n}{4}\log\frac{n}{4} \leq c\, n \log n$, per qualche $c < 1$ ed n abbastanza grande. Ponendo $c = ¾$ otteniamo:

$$3\frac{n}{4} log\, \frac{n}{4} \leq \frac{3}{4}\, n \log n$$

che è vera, quindi $T(n) = \Theta(n \log n)$.

Esempio 5.5

$T(n) = 2T(n/2) + \Theta(n \log n)$

- $a = 2,\ b = 2$
- $f(n) = \Theta(n \log n)$
- $n^{\log_b a} = n^{\log_2 2} = n$

Ora, $f(n) = \Theta(n \log n)$ è asintoticamente più grande di $n^{\log_b a} = n$, ma non polinomialmente più grande. Infatti, $\log n$ è asintoticamente minore di n^{ε} per qualunque valore di $\varepsilon > 0$. Di conseguenza non possiamo applicare il metodo del teorema principale.

Possiamo invece applicare il metodo iterativo:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n \log n) = 2\left(2T\left(\frac{n}{4}\right) + \Theta\left(\frac{n}{2}\log\frac{n}{2}\right)\right) + \Theta(n \log n) = \ldots = 2^k T\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k-1} \Theta\left(n \log \frac{n}{2^i}\right)$$

Ci fermiamo al valore $k = \log n$, quindi otteniamo:

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

$$= n\Theta(1) + \Theta\left(n \sum_{i=0}^{\log n-1} \log n \; - \; n \sum_{i=0}^{\log n-1} \log 2^i\right) =$$

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

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

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

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

5.4.2 Dimostrazione del teorema principale

Dimostreremo il teorema principale solo per valori di *n* che siano potenze esatte, al fine di semplificare i calcoli.

Per una dimostrazione completa si dovrebbe gestire anche il caso di tutti gli altri valori interi, ma ciò richiede di applicare nei conteggi gli operatori di parte intera superiore e inferiore causando notevoli complicazioni senza nulla aggiungere alla sostanza del risultato. Questa parte viene quindi omessa ma può comunque essere consultata sul libro di testo.

Si noti che dimostrare il teorema solo per potenze esatte di per sè non è sufficiente a dimostrare il teorema in generale. Infatti, nel caso di una ipotetica relazione di ricorrenza definita in questo modo:

- $T(n) = n$ se n è una potenza di b
- $T(n) = n^2$ se n non è una potenza di b

La soluzione $T(n) = \Theta(n)$, valida per le potenze di due, non lo sarebbe ovviamente affatto per gli altri valori di *n*.

D'altro canto, nel nostro caso le equazioni di ricorrenza esprimono sempre dei costi computazionali, e quindi sono sempre delle funzioni monotone crescenti in *n*: il caso contrario implicherebbe che la soluzione di un problema di una certa dimensione possa costare meno della soluzione di un problema di dimensione inferiore, il che non è certamente una situazione che si incontra usualmente nella realtà.

Occupiamoci dunque di dimostrare il teorema per l'equazione di ricorrenza:

- $T(n) = aT(n/b) + f(n)$
- $T(1) = \Theta(1)$
- n è una potenza di $b > 1$

Iteriamo la ricorrenza:

$$T(n) = aT(\frac{n}{b}) + f(n) =$$

$$= a\left[aT\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)\right] + f(n) = a^2T\left(\frac{n}{b^2}\right) + a\,f\left(\frac{n}{b}\right) + f(n) =$$

$$= a^2\left[aT\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right)\right] + a\,f\left(\frac{n}{b}\right) + \ f(n) = a^3T\left(\frac{n}{b^3}\right) + a^2f\left(\frac{n}{b^2}\right) + a\,f\left(\frac{n}{b}\right) + f(n) =$$

...

$$= a^k T\left(\frac{n}{b^k}\right) + \sum_{i=0}^{k-1} a^i f\left(\frac{n}{b^i}\right)$$

dove, analogamente a quanto già visto in casi precedenti, l'iterazione termina per il valore *k* tale che $\frac{n}{b^k} = 1$, ossia $k = \log_b n$.

Dunque:

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

Ora,

$$a^{\log_b n} = a^{\log_b a \log_a n} = a^{\log_a n \log_b a} = n^{\log_b a}$$

e quindi l'equazione diviene:

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

Valutiamo la sommatoria nei tre casi del teorema.

Caso 1

$f(n) = O(n^{\log_b a - \varepsilon})$ per qualche costante $\boldsymbol{\varepsilon} > 0$; allora $f\left(\frac{n}{b^i}\right) = O\left(\left(\frac{n}{b^i}\right)^{\log_b a - \varepsilon}\right)$

Quindi:

$$\sum_{i=0}^{\log_b n - 1} a^i f\left(\frac{n}{b^i}\right) = O\left(\sum_{i=0}^{\log_b n - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \varepsilon}\right) =$$

$$= O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n - 1} a^i \left(\frac{1}{b^i}\right)^{\log_b a - \varepsilon}\right) = O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n - 1} \left(\frac{a}{b^{\log_b a - \varepsilon}}\right)^i\right) =$$

$$= O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n - 1} \left(\frac{ab^\varepsilon}{b^{\log_b a}}\right)^i\right) = O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n - 1} \left(\frac{ab^\varepsilon}{a}\right)^i\right) =$$

$$= O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n - 1} (b^\varepsilon)^i\right)$$

Ricordando che la somma della progressione geometrica è $\sum_{i=0}^{k} v^i = \frac{v^{k+1}-1}{v-1}$ abbiamo:

$$O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\log_b n - 1} (b^\varepsilon)^i\right) = O\left(n^{\log_b a - \varepsilon} \frac{(b^\varepsilon)^{\log_b n} - 1}{b^\varepsilon - 1}\right) =$$

$$= O\left(n^{\log_b a - \varepsilon} \frac{\left(b^{\log_b n}\right)^\varepsilon - 1}{b^\varepsilon - 1}\right) = O\left(n^{\log_b a - \varepsilon} \frac{n^\varepsilon - 1}{b^\varepsilon - 1}\right)$$

Ora, b^{ε} è una costante, quindi $\frac{n^{\varepsilon}-1}{b^{\varepsilon}-1} = O(n^{\varepsilon})$ per cui

$$O\left(n^{\log_b a-\varepsilon}\frac{n^{\varepsilon}-1}{b^{\varepsilon}-1}\right)= O\left(n^{\log_b a}\right)$$

L'equazione di ricorrenza iniziale ammette quindi come soluzione:

$$T(n) = \Theta(n^{\log_b a})+ O\left(n^{\log_b a}\right) = \Theta(n^{\log_b a})$$

CVD(1)

Caso 2

$f(n) = \Theta(n^{\log_b a})$; allora $f\left(\frac{n}{b^i}\right) = \Theta\left(\left(\frac{n}{b^i}\right)^{\log_b a}\right)$

Quindi:

$$\sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right) = \Theta\left(\sum_{i=0}^{\log_b n-1} a^i \left(\frac{n}{b^i}\right)^{\log_b a}\right) = \Theta\left(n^{\log_b a}\sum_{i=0}^{\log_b n-1}\left(\frac{a}{b^{\log_b a}}\right)^{i}\right) =$$

$$= \Theta\left(n^{\log_b a}\sum_{i=0}^{\log_b n-1}\left(\frac{a}{a}\right)^{i}\right) = \Theta\left(n^{\log_b a}\sum_{i=0}^{\log_b n-1} 1^i\right) = \Theta\left(n^{\log_b a}\log_b n\right)=$$

$$= \Theta\left(n^{\log_b a}\log_b 2\log_2 n\right)= \Theta\left(n^{\log_b a}\log n\right)$$

L'equazione di ricorrenza ammette quindi come soluzione:

$$T(n) = \Theta(n^{\log_b a})+ \Theta\left(n^{\log_b a}\log n\right) = \Theta\left(n^{\log_b a}\log n\right)$$

CVD(2)

Caso 3

$f(n) = \Omega(n^{\log_b a+\varepsilon})$ per qualche costante $\boldsymbol{\varepsilon} > 0$, $af\left(\frac{n}{b}\right) \leq cf(n)$ per qualche costante $c < 1$ e per n sufficientemente grande.

Dato che $af\left(\frac{n}{b}\right) \leq cf(n)$, abbiamo:

$$f(n) \geq \frac{a}{c}f\left(\frac{n}{b}\right)\geq \frac{a}{c}\left(\frac{a}{c}f\left(\frac{n}{b^2}\right)\right) = \frac{a^2}{c^2}f\left(\frac{n}{b^2}\right)\geq \ldots \geq \frac{a^i}{c^i}f\left(\frac{n}{b^i}\right)$$

Quindi:

$$\sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right) = \sum_{i=0}^{\log_b n-1} c^i\frac{a^i}{c^i}f\left(\frac{n}{b^i}\right) \leq \sum_{i=0}^{\log_b n-1} c^i f(n)= f(n)\sum_{i=0}^{\log_b n-1} c^i$$

Ricordando che quando $|c| < 1$ la somma della serie geometrica è:

$$\sum_{i=0}^{\infty} c^i = \frac{1}{1-c}$$

abbiamo: $\sum_{i=0}^{\log_b n-1} a^i f\left(\frac{n}{b^i}\right) \leq f(n)\sum_{i=0}^{\log_b n-1} c^i < f(n)\sum_{i=0}^{\infty} c^i = f(n)\frac{1}{1-c}$

Da ciò segue che $\sum_{i=0}^{log_b n-1} a^i f\left(\frac{n}{b^i}\right) = O(f(n))$. E' facile vedere inoltre che il primo addendo della sommatoria è proprio *f(n)*, e quindi $\sum_{i=0}^{log_b n-1} a^i f\left(\frac{n}{b^i}\right) = \Omega(f(n))$. Quindi $\sum_{i=0}^{log_b n-1} a^i f\left(\frac{n}{b^i}\right) = \Theta(f(n))$.

Da ciò segue che l'equazione di ricorrenza iniziale diviene:

$$T(n) = \Theta(n^{\log_b a}) + \Theta(f(n))$$

Ma, per l'ipotesi, $f(n) = \Omega(n^{log_b a+\varepsilon})$ e di conseguenza nella relazione precedente il termine dominante è $f(n)$. Dunque,

$$T(n) = \Theta(f(n)).$$

CVD(3)

Concludiamo questo capitolo con un esercizio riepilogativo.

Esempio 5.6

Si risolva la seguente equazione di ricorrenza con tutti e quattro i metodi:

- $T(n)=2T(n/2) + \Theta(n)$
- $T(1)= \Theta(1)$

Per aggirare la difficoltà di "indovinare" la soluzione per il metodo di sostituzione, risolviamo prima tramite **metodo iterativo**:

$$T(n)=2T(n/2)+ \Theta(n)=2(2T(n/2^2)+ \Theta(n/2))+ \Theta(n)=2^2T(n/2^2)+2\,\Theta(n/2)+ \Theta(n)=$$

$$=2^2(2T(n/2^3)+ \Theta(n/2^2))+2\,\Theta(n)=2^3T(n/2^3)+3\,\Theta(n)=\ldots=$$

$$=2^kT(n/2^k)+k\,\Theta(n)=\ldots(\text{finché } n/2^k=1 \text{ cioè se e solo se } k=\log n)\ldots=$$

$$2^{\log n}\,T(1)+\log n\,\Theta(n)=n\,\Theta(1)+\log n\,\Theta(n)=\Theta(n \log n).$$

Dunque la soluzione tramite questo metodo è $\Theta(n \log n)$.

Sfruttiamola per risolvere tramite il **metodo per sostituzione**:

Proviamo a dimostrare per induzione che la soluzione è $\Theta(n \log n)$. Dobbiamo innazitutto eliminare la notazione asintotica, quindi l'equazione diventa:

$T(n)=2T(n/2)+cn$ per qualche costante c>0

$T(1)=d$ per qualche costante d>0

Proviamo a dimostrare per induzione che la soluzione è $T(n) \geq kn \log n$, dove k è una costante da determinare.

Passo base. $T(1) \geq 0$, che è sempre verificata.

Passo induttivo. Sostituendo nell'equazione generica otteniamo:

$$T(n) \geq 2\,(k\,n/2 \log n/2) + cn = kn \log n/2 + cn = kn \log n - kn + cn \geq kn \log n$$

se e solo se $cn \geq kn$ cioè se e solo se $c \geq k$. Poiché un tale k è sempre possibile da trovare, ne concludiamo che $T(n) = \Omega(n \log n)$.

Per quanto riguarda la maggiorazione, non possiamo usare $T(n) \leq k' n \log n$ perché in tal modo il passo base non è verificato. Tentiamo allora $T(n) \leq k' n \log n + h$.

Passo base. $T(1) = d \leq h$ che è vera per opportuni valori di h.

Passo induttivo. Sostitendo l'ipotesi induttiva nell'equazione otteniamo:

$$T(n) \leq 2\,(k' n/2 \log n/2 + h) + cn = k' n \log n/2 + 2h + cn = k'n\,(\log n - 1) + 2h + cn =$$

$$k'n \log n + h - k'n + cn + h \leq k'n \log n + h \text{ se e solo se } cn + h \leq k'n.$$

Anche in questo caso esistono opportuni valori di h e k', fissato c, per cui la disuguaglianza è verificata. Ne segue che $T(n) = O(n \log n)$.

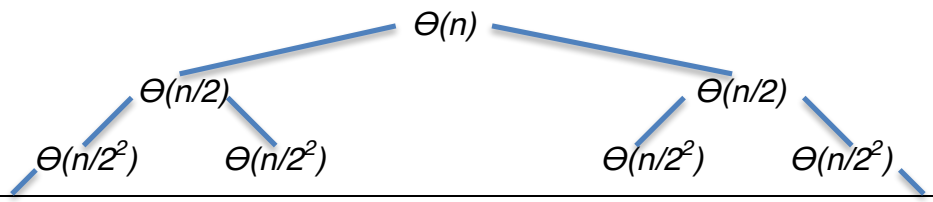
Mettendo insieme le due notazioni asintotiche trovate, si deduce che

$$T(n) = \Theta(n \log n).$$

Osserviamo che il **metodo principale** si può applicare perché la ricorrenza soddisfa le ipotesi del teorema; inoltre $\log_b a = 1$ e quindi si vede facilmente che l'equazione ricade nel caso 2, da cui $T(n) = \Theta(n \log n)$.

Concludiamo con il **metodo dell'albero:**

La radice dell'albero dà un contributo di $\Theta(n)$ ed ha due figli, entrambi etichettati con $T(n/2)$, che quindi danno ciascuno contributo $\Theta(n/2)$ e così via. Schematizziamo dunque l'albero in questo modo:



… … … … … … … … … … … …

T(1)= Θ(1) … … … *T(1)= Θ(1)*

Il contributo della generica riga *i*-esima dell'albero è dato dal valore su ciascuno dei suoi nodi, pari a $\Theta(n/2^i)$, moltiplicato per il numero di nodi, cioè 2^i. Considerato che le righe sono $\log n + 1$ (da 0 a $\log n$), si ha:

$$T(n) = \sum_{i=0}^{\log n} \Theta(\frac{n}{2^i}) 2^i = \sum_{i=0}^{\log n} \Theta(n) = \Theta(n \log n)$$