Relazioni di Ricorrenza e Teorema Fondamentale

Quando un algoritmo contiene una chiamata ricorsiva a se stesso, il suo tempo di esecuzione può spesso essere descritto da un'equazione di ricorrenza, o semplicemente ricorrenza, che descrive il tempo complessivo su un problema di taglia n in termini del suo tempo di esecuzione su input più piccoli.

Una ricorrenza per il tempo di esecuzione di un algoritmo progettato in accordo al paradigma divide-et-impera, può essere ottenuta considerando i tre passi del paradigma.

Sia $T(n)$ il tempo di esecuzione sul problema di taglia n.

Se n è piccolo abbastanza, diciamo $n \leq c$, allora $T(n) = \Theta(1)$.

Viceversa, se dividiamo il problema in a sottoproblemi, ciascuno di taglia $1/b$ della taglia originale, e $D(n)$ indica il tempo per la divisione mentre $C(n)$ il tempo per ri-combinare le soluzioni dei sottoproblemi, si ha:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq c \\ a\,T(n/b) + D(n) + C(n) & \text{altrimenti.} \end{cases}$$

Master theorem. Sia $f(n)$ una funzione non decrescente e siano a, b, n_0, c_0 e c delle costanti tali che $a \geq 1$, $b > 1$ e $n_0, c_0, c > 0$, per la relazione di ricorrenza

$$T(n) \leq \begin{cases} c_0 & \text{se } n \leq n_0 \\ a\,T(n/b) + c \cdot f(n) & \text{altrimenti.} \end{cases}$$

dove n/b va interpretato come $\lfloor n/b \rfloor$ o $\lceil n/b \rceil$.

Se esistono due costanti positive γ e n_0' tali che $a\,f(n/b) \leq \gamma f(n)$ $\forall n \geq n_0'$, allora la relazione di ricorrenza ha i seguenti limiti superiori:

1. $T(n) = O(f(n))$ se $\gamma < 1$
2. $T(n) = O(f(n) \cdot \log_b(n))$ se $\gamma = 1$
3. $T(n) = O(n^{\log_b a})$ se $\gamma > 1$ e $a > 1$

Prima di provare l'enunciato, cerchiamo di capire "cosa dice".

I tre casi dipendono dalla comparazione di

$$a\, f\left(\frac{n}{b}\right) \leq g\, f(n)$$

- a: # sottoproblemi
- b: taglia sottoproblemi
- $f\left(\frac{n}{b}\right)$: costo della suddivisione dei sottoproblemi e della ricombinazione delle soluzioni
- $g\, f(n)$: costo della suddivisione del problema originale e della fusione delle soluzioni dei sottoproblemi

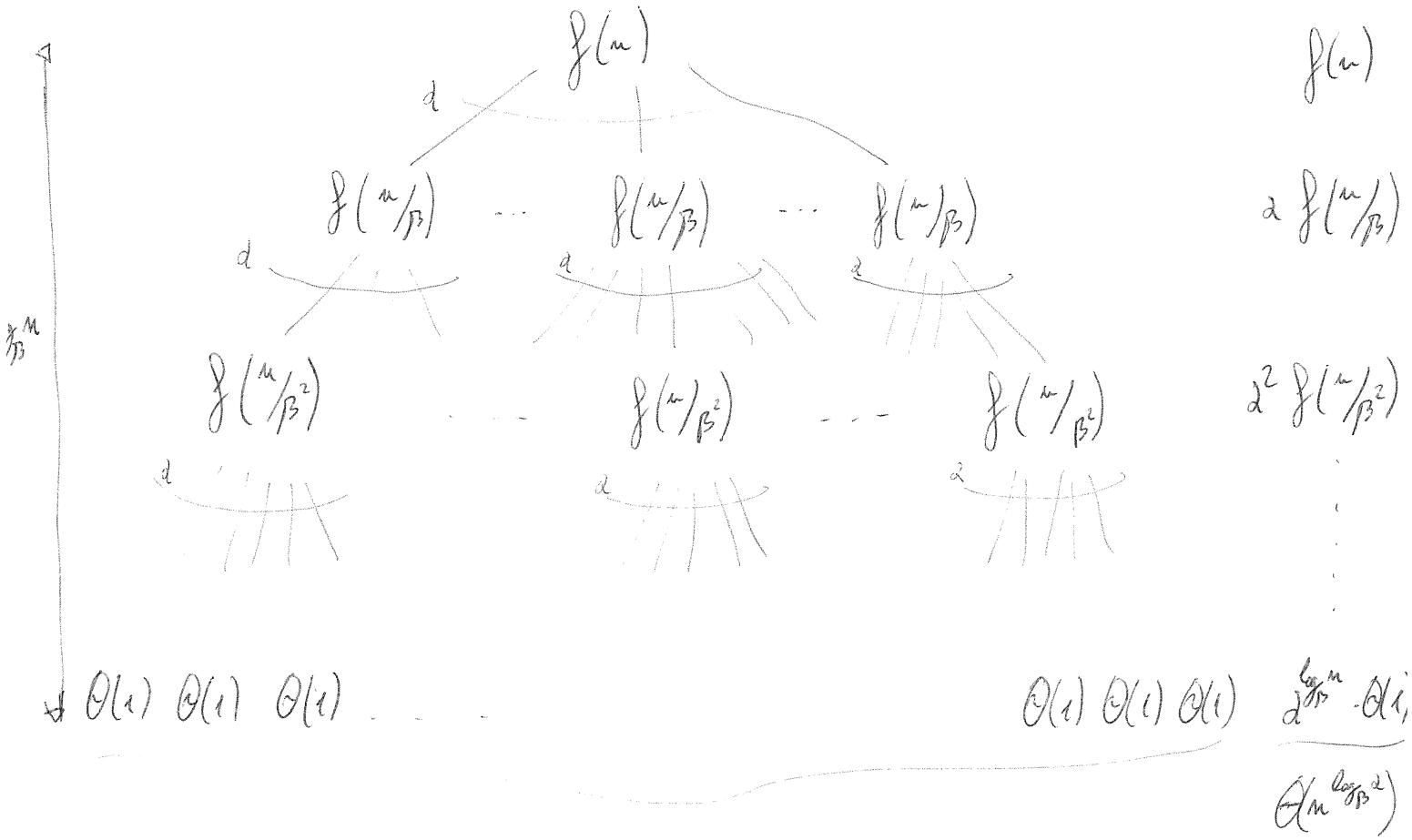
Intuitivamente, quindi, il caso 1. si verifica quando nell'algoritmo in questione il costo di suddivisione/ricombinazione è predominante, è maggiore su input più grandi, e pesa di più delle soluzioni dei sottoproblemi elementari. Pertanto, $T(n) = O(f(n))$.

Il caso 2., invece, si verifica quando il costo di suddivisione/ricombinazione è distribuito allo stesso modo lungo i $\log_b n$ livelli della ricorsione. Pertanto, $T(n) = O(f(n) \cdot \log_b n)$.

Infine, il caso 3. si verifica quando ai livelli "più bassi" della ricorsione i costi di suddivisione/ricombinazione sono maggiori. In tal caso $T(n) = O(n^{\log_b a})$,

cioè il costo predominante è quello di risoluzione dei problemi elementari

Visivamente:



In parole diverse:

Il caso 1. rappresenta le situazioni in cui il costo è dominato dalla radice

Il caso 2. rappresenta le situazioni in cui il costo è equamente distribuito lungo i livelli

Il caso 3. rappresenta le situazioni in cui il costo è dominato dalle foglie.

Dimostrazione. Per dimostrare l'enunciato assumiamo per semplicità che $n_0 = n_0' = 1$ e che n sia una potenza di B.

Si noti che il primo livello della ricorsione contribuisce al costo con $c \cdot f(n)$.

Il secondo con $c \cdot a f(n/B)$. Il terzo con $c \cdot a^2 \cdot f(n/B^2)$, e così via. L'ultimo livello contribuisce con al più $c \cdot a^h \cdot f(n/B^h)$.

↓ alcune chiamate possono essere terminate prima

Quindi, possiamo scrivere

$$T(n) \leq c \cdot f(n) + c \cdot a f(n/B) + \dots + c \cdot a^h \cdot f(n/B^h)$$

$$= c \cdot \sum_{i=0}^{h} a^i \cdot f(n/B^i).$$

Il valore di h può essere determinato osservando che

$$n/B^h = n_0 = 1 \iff h = \log_B n.$$

Inoltre, ragionando per induzione, possiamo provare che:

$$a^i \cdot f(n/B^i) \leq \gamma^i \cdot f(n)$$

Procediamo.

Base dell'induzione. Consideriamo sia $i=0$ che $i=1$ (ci serviranno entrambi)

Per $i=0$ $\quad a^0 \cdot f\left(\frac{n}{b^0}\right) = f(n) \leq \gamma^0 \cdot f(n) = f(n)$.

Per $i=1$ $\quad a\, f\left(\frac{n}{b}\right) \leq \gamma f(n)$ per l'ipotesi del master theorem.

Ipotesi induttiva. Sia $i > 1$. Risulta $a^i f\left(\frac{n}{b^i}\right) \leq \gamma^i f(n)$

Passo induttivo. Osserviamo che $a^{i+1} \cdot f\left(\frac{n}{b^{i+1}}\right)$ può essere scritto come:

$$a^i \left(a \cdot f\left(\frac{\left(\frac{n}{b^i}\right)}{b} \right) \right) \leq \left(\begin{array}{l} \text{considerando} \\ \left(\frac{n}{b^i}\right) = n' \text{ e} \\ \text{applicando la} \\ a \cdot f\left(\frac{n'}{b}\right) \leq \gamma f(n') \end{array} \right) a^i \left(\gamma \cdot f\left(\frac{n}{b^i}\right) \right)$$

$$= \gamma \left(a^i \cdot f\left(\frac{n}{b^i}\right) \right) \leq \left(\begin{array}{l} \text{per l'ipotesi} \\ \text{induttiva} \end{array} \right) \gamma \left(\gamma^i \cdot f(n) \right)$$

$$= \gamma^{i+1} f(n).$$

L'affermazione è pertanto dimostrata.

Discende che:

$$T(n) \leq c \cdot \sum_{i=0}^{h} a^i \cdot f\left(\frac{n}{b^i}\right) \leq c \cdot \sum_{i=0}^{h} \gamma^i \cdot f(n) = c \cdot f(n) \sum_{i=0}^{h} \gamma^i.$$

A questo punto possiamo valutare i 3 casi del Master theorem a seconda del valore di γ. Ricordiamo che:

$$\sum_{i=0}^{h} \gamma^i = \frac{\gamma^{h+1} - 1}{\gamma - 1} \quad \text{per } \gamma \neq 1.$$

1. Nel primo caso risulta $\gamma < 1$, per cui

$$\frac{\gamma^{h+1} - 1}{\gamma - 1} = \frac{1 - \gamma^{h+1}}{1 - \gamma} < \frac{1}{1 - \gamma} = O(1)$$

Pertanto $T(n) \leq c \cdot f(n) \cdot \sum_{i=0}^{h} \gamma^i = c \cdot f(n) \cdot O(1) = O(f(n))$.

2. Nel secondo caso risulta $\gamma = 1$, per cui

$$\sum_{i=0}^{h} \gamma^i = h + 1 = \left(\begin{array}{c}\text{ricordando che} \\ h = \log_\beta n\end{array}\right) = \log_\beta n + 1 = O(\log_\beta n)$$

Pertanto $T(n) \leq c \cdot f(n) \cdot \sum_{i=0}^{h} \gamma^i = c \cdot f(n) \cdot O(\log_\beta n) = O(f(n) \cdot \log_\beta n)$

3. Nel terzo caso risultano $\gamma > 1$ e $\alpha > 1$. Assumendo che valga l'eguaglianza stretta, cioè $\alpha f(n/\beta) = \gamma \cdot f(n)$, risulta $\alpha^i \cdot f(n/\beta^i) = \gamma^i \cdot f(n) \quad \forall i = 0, 1, \ldots$ e, quindi,

$$T(n) \leq c \cdot f(n) \cdot \sum_{i=0}^{h} \gamma^i = c \cdot f(n) \cdot \frac{\gamma^{h+1} - 1}{\gamma - 1} \leq c' \cdot f(n) \cdot \gamma^h$$

$$= c' \cdot \alpha^h \cdot f(n/\beta^h) = O(\alpha^h \cdot f(n/\beta^h))$$

Quindi possiamo scrivere:

$$T(n) = O(a^h \cdot f(n/\beta^h)) = O(a^h \cdot f(n_0)) = O(a^{\log_\beta n})$$

costante "assorbita" in O

$$= O(2^{\log_2 a \cdot \log_\beta n}) = O(2^{\log_2 a \cdot (\log_2 n / \log_2 \beta)})$$

$$= O(2^{(\log_2 a / \log_2 \beta) \cdot \log_2 n}) = O(2^{\log_2 n \cdot (\log_2 a / \log_2 \beta)})$$

$$= O(n^{\log_\beta a}).$$

$\forall a, b, c > 0$

$$\log_b a = \frac{\log_c a}{\log_c b}$$

Pertanto il teorema è dimostrato nel caso in cui n è una potenza di β. Se ciò non accade:

1. $\lfloor n/\beta \rfloor$ può essere maggiorato con n/β e

$$T(n) \leq c \cdot \sum_{i=0}^{h} a^i \cdot f(n/\beta^i)$$ è ancora valida.

La prova funziona ancora

2. $\lceil n/\beta \rceil$ richiede che la condizione del teorema sia

$$a f(\lceil n/\beta \rceil) \leq \gamma \cdot f(n) \quad \forall n \geq n_0.$$

Applicazione alla risoluzione della ricorrenza del MergeSort. La ricorrenza era

$$T(n) \leq \begin{cases} c_0 & \text{se } n \leq 1 \\ 2T(n/2) + c \cdot n & \text{altrimenti} \end{cases}$$

In questo caso: $\alpha = 2$, $\beta = 2$ e $f(n) = n$

la condizione:

$$\alpha f(n/\beta) \leq \gamma f(n)$$

diventa:

$$2 \cdot n/2 \leq \gamma \cdot n,$$

da fissando $\gamma = 1$, risulta soddisfatta $\forall n \geq n_0' = 1$

$\gamma = 1$: costante positiva.

Ricadiamo, quindi, nel caso 2. del Master Theorem

e

$$T(n) = O(n \cdot \log_2 n).$$

Nota: il Master Theorem, quando applicabile, è uno strumento utilissimo per risolvere immediatamente relazioni di ricorrenza.

Dopo aver rivisto e fatti propri i passi della dimostrazione del Master Theorem, applicando passi simili, per esercizio si dimostri la seguente "specializzazione" del Master theorem.

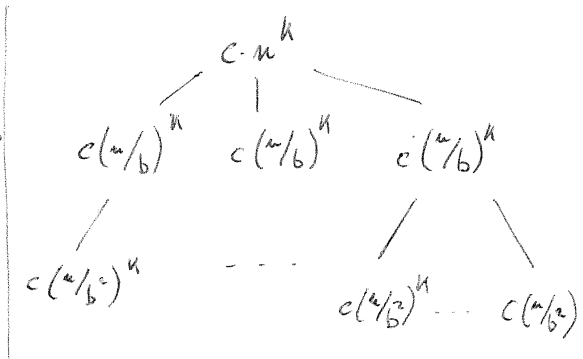
Esercizio. La ricorrenza

$$T(n) \leq \begin{cases} c_0 & \text{se } n \leq n_0 \\ a\,T(n/b) + c \cdot n^K & \text{altrimenti} \end{cases}$$

dove a, b, c e K sono costanti positive, la soluzioni

1. $T(n) = O(n^K)$ se $a < b^K$
2. $T(n) = O(n^K \cdot \log n)$ se $a = b^K$
3. $T(n) = O(n^{\log_b a})$ se $a > b^K$

Suggerimento. Si sommino tutti i termini che contribuiscono al costo della ricorrenza.



Dalla somma si evidensi il termine comune $c \cdot n^K$, riscrivendola come $c \cdot n^K \cdot [\ \ldots\ ]$

Si ponga $q = a/b^K$ e si considerino i tre casi risultanti ...

Esercizi. Risolvere le ricorrenze che seguono:

a) $$T(n) \leq \begin{cases} c & n \leq n_0 \\ 7T(n/5) + cn & \text{altrimenti} \end{cases}$$

b) $$T(n) \leq \begin{cases} c & n \leq n_0 \\ 2T(n/3) + cn & \text{altrimenti} \end{cases}$$

c) $$T(n) \leq \begin{cases} c & n \leq n_0 \\ 7T(n/5) + cn^2 & \text{altrimenti} \end{cases}$$

d) $$T(n) \leq \begin{cases} c & n \leq n_0 \\ 2T(n/3) + cn^2 & \text{altrimenti} \end{cases}$$

e) $$T(n) \leq \begin{cases} c & n \leq n_0 \\ 5T(n/4) + cn & \text{altrimenti} \end{cases}$$

f) $$T(n) \leq \begin{cases} c & n \leq n_0 \\ 3T(n/3) + cn^3 & \text{altrimenti} \end{cases}$$