Appunti lezione – Capitolo 2
Analisi delle funzioni di costo

Alberto Montresor

20 Settembre, 2016

1 Funzioni di costo

Definizione 1 (Funzione di costo)**.** Utilizziamo il termine *funzione di costo* per indicare una funzione $f : \mathbb{N} \to \mathbb{R}$ dall'insieme dei numeri naturali ai reali, asintoticamente positiva.

Come possiamo confrontare le funzioni di costo che abbiamo ottenuto finora?

- Insertion Sort, caso ottimo: $f(n) = a_1 n + b_1$
- Insertion Sort, caso pessimo-medio: $f(n) = a_2 n^2 + b_2 n + c_2$
- Merge Sort (sempre): $f(n) = a_3 n \log n + b_3 n + c_3$

Ci sono troppi dettagli: termini di vari ordini, costanti moltiplicative. In realtà: siamo interessati solo al "tasso di crescita" di queste funzioni ovvero, al loro comportamento asintotico.

Consideriamo queste coppie di funzioni di costo; quali sono preferibili?

- $\log_2 n$ oppure $\log_{10} n$
- $\log n^{1000}$ oppure $\log n$
- $\log^{1000} n$ oppure $n^{0.001}$
- $1000n$ oppure $0.001 \cdot n^2$
- n^{1000} oppure $0.001 \cdot 2^n$

Vediamo come crescono vari tipi di funzione al crescere di n:

$f(n)$	10	100	1000	10000	tipo
$\log n$	3	6	9	13	logaritmico
$\sqrt{n}$	3	10	31	100	sublineare (polinomiale)
n	10	100	1000	10000	lineare (polinomiale)
$n \log n$	30	664	9965	132877	loglineare (polinomiale)
n^2	10^2	10^4	10^6	10^8	quadratico (polinomiale)
n^2	10^3	10^6	10^9	10^{12}	cubico (polinomiale)
2^n	1024	10^{30}	10^{300}	10^{3000}	esponenziale

Figura 1: Esempi di valori di complessità

2 Notazioni O, Ω, Θ

Siamo interessati a caratterizzare la crescita delle funzioni in base al loro comportamento *asintotico*, ovvero al crescere del valore di ingresso (dimensione del problema).

Definizione 2 (Notazione O)**.** Sia $g(n)$ una funzione di costo; indichiamo con $O(g(n))$ l'insieme delle funzioni $f(n)$ tali per cui:

$$\exists c > 0, \exists m \geq 0 : 0 \leq f(n) \leq cg(n), \forall n \geq m.$$

In altre parole:

- asintoticamente la funzione giace sotto $cg(n)$;
- $f(n)$ cresce al più come $g(n)$;
- $g(n)$ è un limite **asintotico** superiore per $f(n)$.

Definizione 3 (Notazione Ω)**.** Sia $g(n)$ una funzione di costo; indichiamo con $\Omega(g(n))$ l'insieme delle funzioni $f(n)$ tali per cui:

$$\exists c > 0, \exists m \geq 0 : 0 \leq cg(n) \leq f(n), \forall n \geq m.$$

In altre parole:

- asintoticamente la funzione giace sopra $cg(n)$;
- $f(n)$ cresce almeno quanto $g(n)$;
- $g(n)$ è un limite **asintotico** inferiore per $f(n)$.

Definizione 4 (Notazione Θ)**.** Sia $g(n)$ una funzione di costo; indichiamo con $\Theta(g(n))$ l'insieme delle funzioni $f(n)$ tali che per cui:

$$\exists c_1 > 0, \exists c_2 > 0, \exists m \geq 0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq m.$$

In altre parole $f(n) = \Theta(g(n))$ se e solo se $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$.

Abusiamo delle notazioni appena introdotte scrivendo:

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

anche se $O(g(n))$ in realtà è un insieme e dovremmo scrivere

$$f(n) \in O(g(n)), f(n) \in \Omega(g(n)), f(n) \in \Theta(g(n))$$

Diciamo "$f(n)$ è un O grande, Ω, Θ di $g(n)$".

La figura 2 illustra graficamente il significato delle notazioni appena introdotte.

La notazione O a volte è confusa con la notazione Θ; un uso che deriva dalla letteratura passata sull'argomento. Infatti può essere utilizzata più semplicemente della notazione Θ per descrivere la complessità di un algoritmo: se utilizzata per descrivere il caso peggiore, limita il tempo massimo necessario per eseguire un algoritmo.

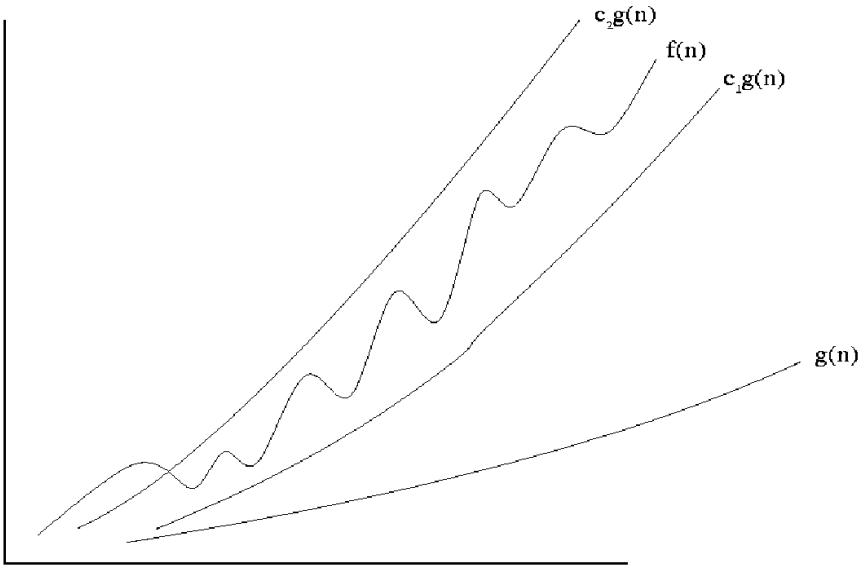

Figura 2: Comportamento asintotico della notazione O

3 Esercizi e proprietà

3.1 Vero o falso: $10n^3 + 2n^2 + 7 = O(n^3)$?

Si consideri la seguente funzione cubica: $f(n) = 10n^3 + 2n^2 + 7$. Vogliamo provare che $f(n) = O(n^3)$. Dobbiamo provare che:

$$\exists c > 0, m \geq 0 : 10n^3 + 2n^2 + 7 \leq cn^3, \forall n \geq m$$

Si può procedere così:

$$\begin{aligned} f(n) &= 10n^3 + 2n^2 + 7 \\ &\leq 10n^3 + 2n^3 + 7 \\ &\leq 10n^3 + 2n^3 + n^3 \qquad\qquad \forall n \geq \sqrt[3]{7} \\ &= 13n^3 \\ &\leq cn^3 \end{aligned}$$

che è vera per $c \geq 13$ e $m = 2$ (utilizziamo un numero intero perchè (i) stiamo parlando di dimensione dell'input, (ii) basta trovare un qualunque valore che soddisfa il limite e (iii) è più facile da maneggare).

3.2 Vero o falso: $3n^2 + 7n = \Theta(n^2)$?

Si consideri la seguente funzione quadratica: $f(n) = 3n^2 + 7n$. Vogliamo provare che $f(n) = \Theta(n^2)$. Dobbiamo provare che:

$$\exists c_1 > 0, c_2 > 0, \exists m \geq 0 : c_1 n^2 \leq 3n^2 + 7n \leq c_2 n^2, \forall n \geq m$$

Si può procedere così:

$$\begin{aligned} f(n) &= 3n^2 + 7n \\ &\leq 3n^2 + 7n^2 && \forall n \geq 1 \\ &= 10n^2 \\ &\leq c_2 n^2 \end{aligned}$$

che è vera per $m = 1$ e $c_2 \geq 10$. Inoltre,

$$\begin{aligned} f(n) &= 3n^2 + 7n \\ &\geq 3n^2 && \forall n \geq 0 \\ &\geq c_1 n^2 \end{aligned}$$

che è vera per $m = 0$ e $c_1 \leq 3$.

3.3 Espressioni polinomiali

Abbiamo visto due espressioni polinomiali. Cerchiamo di trarne una regola generale. Vogliamo provare che un polinomio asintoticamente positivo di grado k cresce come n^k:

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0, a_k > 0 \Rightarrow f(n) = \Theta(n^k)$$

Dobbiamo innanzitutto provare che $f(n) = O(n^k)$, ovvero che

$$\exists c_2 > 0, \exists m \geq n : f(n) \leq c_2 n^k, \forall n \geq m$$

Dimostriamolo:

$$\begin{aligned} f(n) &= a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 \\ &\leq a_k n^k + |a_{k-1}| n^{k-1} + \ldots + |a_1| n + |a_0| \\ &\leq a_k n^k + |a_{k-1}| n^k + \ldots + |a_1| n^k + |a_0| n^k && \forall n \geq 1 \\ &= (a_k + |a_{k-1}| + \ldots + |a_1| + |a_0|) n^k \\ &\leq c_2 n^k \end{aligned}$$

L'ultima relazione è vera per $c_2 \geq (|a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|)$ e per $m = 1$.

Dobbiamo poi provare che $f(n) = \Omega(n^k)$, ovvero che

$$\exists c_1 > 0, \exists m \geq n : f(n) \geq c_1 n^k, \forall n \geq m$$

E' facile notare che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 \geq c n^k \Leftrightarrow c \leq a_k + \frac{a_{k-1}}{n} + \frac{a_{k-2}}{n^2} + \ldots + \frac{a_0}{n^k} > 0$$

In altre parole, se troviamo un valore di n per cui la parte centrale della disequazione è maggiore di 0, lo possiamo scegliere come valore di c.

$$\begin{aligned} a_k + \frac{a_{k-1}}{n} + \frac{a_{k-2}}{n^2} + \ldots + \frac{a_0}{n^k} &\geq \\ a_k - \frac{|a_{k-1}|}{n} - \frac{|a_{k-2}|}{n^2} - \ldots - \frac{|a_0|}{n^k} &\geq && \text{Mettiamo il segno negativo a tutti i coefficienti} \\ a_k - \frac{|a_{k-1}|}{n} - \frac{|a_{k-2}|}{n} - \ldots - \frac{|a_0|}{n} &> 0 && \text{Rimuoviamo l'esponente} \end{aligned}$$

L'ultima espressione è vera se

$$n > \frac{|a_{k-1}| + \ldots + |a_0|}{a_k}$$

3.4 Vero o falso: $6n^2 = O(n)$?

Dobbiamo dimostrare che:

$$\exists c > 0, \exists m \geq 0 : 6n^2 \leq cn, \forall n \geq m$$

Purtroppo abbiamo che:

$$6n^2 \leq cn \Leftrightarrow c \geq 6n$$

Questo significa che c cresce al crescere di n, ovvero che non esiste una "costante" c.

3.5 Vero o falso: $6n^2 = \Omega(n^3)$?

Dobbiamo dimostrare che

$$\exists c > 0, \exists m \geq 0 : cn^3 \leq 6n^2$$

che implica che $c \leq 6/n$; ovvero, qualsiasi sia il valore di c, esisterà un valore di n per cui la disequazione non è soddisfatta.

3.6 Qual è la complessità di $f(n) = 5$?

$$5 = \Theta(n^0) = \Theta(1)$$

3.7 Qual è la complessità di $f(n) = 5 + \sin(n)$?

Il seno è compreso fra -1 e 1; quindi comunque il valore può essere limitato superiormente e inferiormente da una una costante, e la complessità è $\Theta(1)$.

3.8 Logaritmo vs lineare

Vogliamo provare che $\log n = O(n)$. Dobbiamo dimostrare che

$$\exists c > 0, \exists m \geq 0 : \log n \leq cn, \forall n \geq m;$$

La dimostrazione è per induzione su n:

- per $n = 1$, $\log 1 = 0 \leq c$ per qualunque valore positivo di c;
- in generale, per $n \geq 1$:

$$\begin{aligned}
\log(n+1) &\leq \log(n+n) && \text{maggiorazione}, \forall n \geq 1 \\
&= \log 2n \\
&= \log 2 + \log n \\
&= 1 + \log n \\
&\leq 1 + cn && \text{per ipotesi induttiva} \\
&\leq c + cn, c \geq 1 \\
&= c(n+1), c \geq 1
\end{aligned}$$

Abbiamo quindi provato che $\log n \leq cn$ per qualsiasi $c \geq 1, n \geq 1$.

3.9 E se il logaritmo non è in base 2?

Non cambia nulla. Sappiamo infatti che $\log_a n = (\log_a 2)(\log_2 n)$, per $a \neq 1$; inoltre abbiamo già provato che $\log n \leq cn$, per opportuna costante c, pertanto:

$$\log_a n = (\log_a 2)(\log_2 n) \leq (\log_a 2)cn = dn \Leftrightarrow \log_a n = O(n)$$

con $d = c \log_a 2$.

3.10 E se invece di $\log n$ abbiamo $\log n^a, a > 1$?

Non cambia niente. Sappiamo che $\log n^a = a \log n$ e $\log n \leq cn$, per opportuna costante $c > 0$, quindi:

$$\log n^a = a \log n \leq acn \Leftrightarrow \log n^a = O(n)$$

3.11 Polilogaritmi contro polinomi

Generalizziamo; vogliamo dimostrare che per qualsiasi costante $a, b, k > 0$ vale $\log^a n^b = O(n^k)$.
Occorre provare che per ogni $a, b, k > 0$,

$$\exists c > 0, \exists m \geq 0 : (\log^a n^b) \leq cn^k, \forall n \geq m$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che:

$$\log x^y = y \log x$$
$$\log x \leq dx, \forall n \geq 1, d \geq 1$$

Pertanto:

$$\log n^b = b \log n = b\frac{1}{k} k \log n = b\frac{1}{k} \log n^k \leq bd\frac{1}{k} n^k$$

dove $c = bd\frac{1}{k}$.
Per il caso generale:

$$(\log n^b)^a \leq (cn^{k/a})^a = c^a n^k$$

(notare che abbiamo scelto di limitare il contenuto della parentesi con uno specifico valore di k).

3.12 Polinomi vs esponenziali

Vogliamo provare che $n^k = O(a^n), \forall k > 0, \forall a > 1$. Seguiamo la dimostrazione del libro, che si basa su questa proprietà:

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0$$

per tutte le costanti a, b con $a > 1$.
Come vedremo introducendo la notazione o (o-piccolo), questo prova la nostra ipotesi: $n^k = O(a^n), \forall k \geq 0, a > 1$

3.13 Esponenziali con base diversa

Qual è il rapporto fra 2^n e 3^n? Ovviamente $2^n = O(3^n)$, ma è vero anche il contrario $3^n = O(2^n)$?

$$3^n = \left(\frac{3}{2} \cdot 2\right)^n = \left(\frac{3}{2}\right)^n 2^n \leq c2^n$$

Quindi $c \geq \left(\frac{3}{2}\right)^n$, ovviamente impossibile.

3.14 Esponenziale: 2^{n+1}

2^{n+1} è $\Theta(2^n)$ se e solo se $\exists c_1 \geq 0, \exists c_2 \geq 0, m > 0$ tale che

$$c_1 2^n \leq 2^{n+1} \leq c_2 2^n, \forall n \geq m$$

E' facile vedere che $2^{n+1} = 2 \cdot 2^n$, quindi $c_1 \leq 2, c_2 \geq 2, m = 1$.

3.15 Esponenziale: 2^{2n}

2^{2n} è $O(2^n)$ implica $2^{2n} = 2^n \cdot 2^n \leq c2^n$, quindi $c \geq 2^n$, il che è ovviamente assurdo perché non esiste una c costante.

4 Notazione o e ω

Abbiamo dimostrato che $\log^a n = O(n^k)$, ma è vero anche il contrario, ovvero $n^k = O(\log^a n)$? La risposta è no; introduciamo una nuova notazione per descrivere questo fatto.

Definizione 5 (Notazione o)**.** Sia $g(n)$ una funzione di costo; indichiamo con $o(g(n))$ l'insieme delle funzioni $f(n)$ tali per cui:

$$\forall c > 0, \exists m \geq 0 : f(n) < cg(n), \forall n \geq m.$$

Definizione 6 (Notazione ω)**.** Sia $g(n)$ una funzione di costo; indichiamo con $\omega(g(n))$ l'insieme delle funzioni $f(n)$ tali per cui: tali che

$$\forall c > 0, \exists m \geq 0 : cg(n) < f(n), \forall n \geq m.$$

Queste due notazioni vengono utilizzate per definire limiti "stretti". Si noti la differenza fra o e O, oppure fra ω e Ω: la relazione deve valere per tutte le possibili costanti c. Intuitivamente, nella notazione o la funzione $f(n)$ diventa insignificante rispetto a $g(n)$ quando n tende all'infinito; ovvero

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

Si noti che:

$$\begin{aligned} f(n) = o(g(n)) &\Rightarrow f(n) = O(g(n)) \\ f(n) = \omega(g(n)) &\Rightarrow f(n) = \Omega(g(n)) \end{aligned}$$

ma non vangono le implicazioni inverse.
Utilizzando il concetto di limite, date due funzioni $f(n)$ e $g(n)$ si possono fare le seguenti affermazioni:

- se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, allora $f(n) = o(g(n))$;
- se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \neq 0$, allora $f(n) = \Theta(g(n))$;
- se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, allora $f(n) = \omega(g(n))$;

5 Classificazione delle funzioni

E' possibile trarre un'ordinamento di tutte le funzioni, estendendo le relazioni che abbiamo dimostrato fino ad ora: Per qualsiasi $r < s, h < k, a < b$:

$$O(1) \subset O(\log^r n) \subset O(\log^s n) \subset O(n^h) \subset O(n^h \log^r n) \subset O(n^h \log^s n) \subset O(n^k) \subset O(a^n) \subset O(b^n)$$

6 Alcune utili regole

6.1 Dualità

$$\begin{aligned} f(n) = O(g(n)) &\Leftrightarrow g(n) = \Omega(f(n)) \\ f(n) = o(g(n)) &\Leftrightarrow g(n) = \omega(f(n)) \end{aligned}$$

Dimostriamo la prima delle due:

$$\begin{aligned} f(n) = O(g(n)) &\Leftrightarrow \exists c \geq 0, \exists m > 0 : f(n) \leq c \cdot g(n), \forall n \geq m \\ &\Leftrightarrow \exists c \geq 0, \exists m > 0 : g(n) \geq (1/c) \cdot f(n), \forall n \geq m \\ &\Leftrightarrow \exists d \geq 0, \exists m > 0 : g(n) \geq d \cdot f(n), \forall n \geq m \\ &\Leftrightarrow g(n) = \Omega(f(n)) \end{aligned}$$

dove $d = 1/c$. Questa regola ci permette di semplificare le dimostrazioni di molte proprietà.

6.2 Eliminazione delle costanti

$$f(n) = O(g(n)) \Leftrightarrow af(n) = O(g(n)), \forall a > 0$$

Dimostriamo tale relazione:

$$\begin{aligned} f(n) = O(g(n)) &\Leftrightarrow \exists c > 0, \exists m \geq 0 : f(n) \leq cg(n), \forall n \geq m \\ &\Leftrightarrow \exists c > 0, \exists m \geq 0 : af(n) \leq (ac) \cdot g(n), \forall n \geq m \\ &\Leftrightarrow \exists d > 0, \exists m \geq 0 : af(n) \leq dg(n), \forall n \geq m \\ &\Leftrightarrow af(n) = O(g(n)) \end{aligned}$$

con $d = (ac)$. Naturalmente la proprietà vale anche per le notazioni $\Omega, \Theta, o, \omega$

6.3 Sommatoria

$$f_1(n) = O(g_1(n)), f_2(n) = O(g_2(n)) \Rightarrow f_1(n) + f_2(n) = O(g_1(n) + g_2(n))$$

Infatti,

$$\begin{aligned} f_1(n) = O(g_1(n)) \wedge f_2(n) = O(g_2(n)) &\Rightarrow \\ f_1(n) \leq c_1 g_1(n) \wedge f_2(n) \leq c_2 g_2(n) &\Rightarrow \\ f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n) &\Rightarrow \\ f_1(n) + f_2(n) \leq \max\{c_1, c_2\}(g_1(n) + g_2(n)) & \end{aligned}$$

Nota: per ragioni di semplicità, sono stati omessi tutti i quantificatori esistenziali e universali. In realtà, dobbiamo selezionare un opportuno valore m prendendo il massimo fra i valori m_1, m_2 delle due funzioni. Naturalmente la proprietà vale anche per le notazioni $\Omega, \Theta, o, \omega$.

6.4 Prodotto

$$f_1(n) = O(g_1(n)), f_2(n) = O(g_2(n)) \Rightarrow f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$$

Infatti,

$$\begin{aligned} f_1(n) = O(g_1(n)) \wedge f_2(n) = O(g_2(n)) &\Rightarrow \\ f_1(n) \leq c_1 g_1(n) \wedge f_2(n) \leq c_2 g_2(n) &\Rightarrow \\ f_1(n) \cdot f_2(n) \leq c_1 c_2 g_1(n) g_2(n) & \end{aligned}$$

Nota: per ragioni di semplicità, sono stati omessi tutti i quantificatori esistenziali e universali. In realtà, dobbiamo selezionare un opportuno valore m prendendo il massimo fra i valori m_1, m_2 delle due funzioni. Naturalmente la proprietà vale anche per le notazioni $\Omega, \Theta, o, \omega$.

6.5 Simmetria

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

Grazie alla proprietà di dualità:

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

6.6 Transitività

$$f(n) = O(g(n) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$

Infatti:

$$\begin{aligned} f(n) = O(g(n) \wedge g(n) = O(h(n)) &\Rightarrow \\ f(n) \leq c_1 g(n) \wedge g(n) \leq c_2 h(n) &\Rightarrow \\ f(n) \leq c_1 c_2 h(n) & \end{aligned}$$

Nota: per ragioni di semplicità, sono stati omessi tutti i quantificatori esistenziali e universali. In realtà, dobbiamo selezionare un opportuno valore m prendendo il massimo fra i valori m_1, m_2 delle due funzioni. Naturalmente la proprietà vale anche per le notazioni $\Omega, \Theta, o, \omega$.

7 Problemi vs Algoritmi

Complessità in tempo di un algoritmo: *La più grande quantità di tempo richiesta per un input di dimensione n*

- $O(f(n))$: Dimostrare che per ogni input di dimensione n, l'algoritmo richiede non più di $c \cdot f(n)$ per qualche c; ovvero, caso pessimo.
- $\Omega(f(n))$: Trovare un input di dimensione n, tale per cui l'algoritmo richiede almeno $c \cdot f(n)$
- $\Theta(f(n))$: Entrambe

Complessità in tempo di un problema computazionale: *La complessità in tempo dell'algoritmo più veloce che risolve tale problema*

- $O(f(n))$: Trovare un algoritmo che risolva il problema in tempo $O(f(n))$
- $\Omega(f(n))$: Dimostrare che nessun algoritmo può risolvere il problema più velocemente di $\Omega(f(n))$
- $\Theta(f(n))$: Entrambe

Alcune affermazioni in tal senso:

- Il tempo di esecuzione di Insertion Sort è compreso fra $\Omega(n)$ e $O(n^2)$.
- La complessità di Insertion Sort non è $\Omega(n^2)$.
- La complessità di Insertion Sort nel caso peggiore è $\Omega(n^2)$.
- La complessità di Merge Sort è $\Theta(n \log n)$.
- La complessità del problema dell'ordinamento è $O(n \log n)$.
- Che dire sui limiti inferiori del problema dell'ordinamento?

8 Equazioni di ricorrenza

Nel caso degli algoritmi ricorsivi, il tempo di esecuzione può essere descritto da una **equazione di ricorrenza**, ovvero un'equazione che descrive una funzione in termini del suo valore con input più piccoli. Esempio di Merge Sort:

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

Esistono (almeno) tre metodi per risolvere le equazioni di ricorrenza:

- **Metodo di sostituzione, o per tentativi**: Ipotizziamo una soluzione e utilizziamo l'induzione matematica per dimostrare che la nostra ipotesi è corretta.
- **Metodo dell'albero di ricorsione, o per livelli**: Converte la ricorrenza in un albero i cui nodi rappresentano i costi ai vari livelli della ricorsione
- **Metodo dell'esperto (Master Theorem), o delle ricorrenze comuni**: Fornisce una soluzione per ricorrenze della forma: $T(n) = aT(N/b) + f(n)$ con $a > 0, b > 1$

Alcuni dettagli tecnici:

- Il valore di n è sempre un intero. Spesso però scriviamo le equazioni ignorando questo fatto. Esempio: la vera funzione di MergeSort è:

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

- Condizioni al contorno: cosa succede per $n = 1$? Normalmente "dimentichiamo" questo caso. Ma bisogna essere sicuri di ciò che si fa . . . Nel seguito, verificheremo i vari casi.

8.1 Metodo dell'iterazione o dell'albero di ricorsione

8.1.1 $T(n) = T(n/2) + b$

Questa è la funzione di ricorrenza della ricerca binaria:

$$T(n) = \begin{cases} T(n/2) + b & n > 1 \\ 1 & n \leq 1 \end{cases}$$

E' possibile risolverla in questa maniera:

$$\begin{aligned} T(n) &= b + T(n/2) \\ &= b + b + T(n/4) \\ &= b + b + b + T(n/8) \\ &= b + b + b + \ldots + T(1) \end{aligned}$$

$n/2^i = 1$ quando $i = \log n$; quindi $T(n) = b \log n + T(1) = O(\log n)$.

8.1.2 $T(n) = 4T(n/2) + n$

$$T(n) = \begin{cases} 4T(n/2) + n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Normalmente, applicare questo metodo non è così semplice. Ci vogliono una serie di manipolazioni algebriche, come quelle mostrate qui sotto.

$$
\begin{aligned}
T(n) &= n + 4T(n/2) \\
&= n + 4n/2 + 16T(n/2^2) \\
&= n + 2n + 16n/4 + 64T(n/8) \\
&= \dots \\
&= n + 2n + 4n + 8n + \dots + 2^{\log n - 1}n + 4^{\log n}T(1) \\
&= n \sum_{j=0}^{\log n - 1} 2^j + 4^{\log n}
\end{aligned}
$$

Infatti, $n/2^i = 1$ quando $i = \log n$.
Abbiamo due termini, che trattiamo separatamente. Innanzittuto sfruttiamo questa formula:

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

Quindi scriviamo:

$$4^{\log_2 n} = n^{\log_2 4} = n^2$$

Che e' una serie geometrica; applicando la nota formula alla prima parte dell'espressione, si ottiene

$$n \sum_{j=0}^{\log n - 1} 2^j = n\frac{2^{\log n} - 1}{2 - 1} = n(n-1)$$

Quindi il costo totale dell'algoritmo è $\Theta(n^2)$.

Formula Serie geometrica finita

$$\forall x \neq 1 : \sum_{j=0}^{k} x^j = \frac{x^{k+1} - 1}{x - 1}$$

8.1.3 $T(n) = 4T(n/2) + n^3$

Proviamo a risolvere tramite il metodo dell'albero della ricorrenza. Per visualizzare cosa succede, sviluppiamo qui tre livelli dell'albero di ricorsione:

$$
\overbrace{
\overbrace{\left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3}^{\left(\frac{n}{2}\right)^3}
\overbrace{\left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3}^{\left(\frac{n}{2}\right)^3}
\overbrace{\left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3}^{\left(\frac{n}{2}\right)^3}
\overbrace{\left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3}^{\left(\frac{n}{2}\right)^3}
}^{n^3}
$$

In altre parole, la chiamata a livello 0 genera un costo di n^3 e genera 4 chiamate di livello 1, ognuna delle quali ha come dimensione $n/2$ e genera un costo pari a $(n/2)^3$. Ognuna di esse genererà 4 chiamate di livello 2, ognuna delle quali ha come dimensione $(n/4)$ e genera un costo pari a $(n/4)^3$
Sviluppare alberi così larghi non è semplice. E' possibile calcolare il costo di ogni livello in forma tabellare, esplicitando il livello di ricorsione, la dimensione dell'input, il costo di una singola chiamata su quella dimensione (senza considerare le sottochiamate ricorsive), il numero di chiamate effettuate a quel particolare livello, e quindi il costo totale del livello.

Livello	Dimensione	Costo chiamata	N. chiamate	Costo livello
0	n	n^3	1	n^3
1	$n/2$	$(n/2)^3$	4	$4(n/2)^3$
2	$n/4$	$(n/4)^3$	16	$16(n/4)^3$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
i	$n/2^i$	$(n/2^i)^3$	4^i	$4^i(n/2^i)^3$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$\ell-1$	$n/2^{\ell-1}$	$(n/2^{\ell-1})^3$	$4^{\ell-1}$	$4^{\ell-1}(n/2^{\ell-1})^3$
$\ell=\log n$	1	$T(1)$	$4^{\log n}$	$4^{\log n}$

Ancora una volta, il numero di livelli di ricorsione è pari $\ell = \log n$, assumendo per semplicità che n sia una potenza di 2.
Come nell'esempio precedente, l'ultima riga costa $T(1) \cdot 4^{\log n} = n^2$. Sommando i costi per livello di tutte le righe, si ottiene:

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log n-1} 4^i \cdot n^3/2^{3i} + n^2 \\
&= n^3 \sum_{i=0}^{\log n-1} \frac{2^{2i}}{2^{3i}} + n^2 \\
&= n^3 \sum_{i=0}^{\log n-1} \left(\frac{1}{2}\right)^i + n^2 \\
&\le n^3 \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i + n^2 && \text{Estensione della sommatoria} \\
&= \frac{1}{1-\frac{1}{2}} n^3 + n^2 && \text{Serie geometrica}
\end{aligned}
$$

da cui si ottiene:

$$T(n) \le n^2 + 2n^3 = O(n^3)$$

Si noti che avendo esteso la sommatoria, abbiamo introdutto una disuguaglianza nella catena delle formule. Per questo motivo, non possiamo scrivere che $T(n) = \Theta(n^3)$, in quanto abbiamo solo dimostrato che $T(n)$ è dominato superoriormente da una funzione $O(n^3)$.
La formula utilizza è la seguente

Formula Serie geometrica infinita decrescente

$$\forall x, |x| < 1 : \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$$

8.1.4 $T(n) = 4T(n/2) + n^2$

Utilizzando il metodo dell'albero di ricorrenza, i primi tre livelli sono molto simili:

$$\overbrace{\left(\frac{n}{2}\right)^2 \qquad \left(\frac{n}{2}\right)^2 \qquad \left(\frac{n}{2}\right)^2 \qquad \left(\frac{n}{2}\right)^2}^{n^2}$$

$$\overbrace{\left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2}^{\left(\frac{n}{2}\right)^2} \overbrace{\left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2}^{\left(\frac{n}{2}\right)^2} \overbrace{\left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2}^{\left(\frac{n}{2}\right)^2} \overbrace{\left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2 \left(\frac{n}{4}\right)^2}^{\left(\frac{n}{2}\right)^2}$$

Come sopra, esplicitiamo il costo dei livelli tramite una tabella:

Livello	**Dimensione**	**Costo chiamata**	**N. chiamate**	**Costo livello**
0	n	n^2	1	n^2
1	$n/2$	$(n/2)^2$	4	$4(n/2)^2$
2	$n/4$	$(n/4)^2$	16	$16(n/4)^2$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
i	$n/2^i$	$(n/2^i)^2$	4^i	$4^i(n/2^i)^2$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$\ell-1$	$n/2^{\ell-1}$	$(n/2^{\ell-1})^2$	$4^{\ell-1}$	$4^{\ell-1}(n/2^{\ell-1})^2$
$\ell = \log n$	1	$T(1)$	$4^{\log n}$	$4^{\log n}$

Ancora una volta, il numero di livelli di ricorsione è pari $\ell = \log n$, assumendo per semplicità che n sia una potenza di 2.
Come nell'esempio precedente, l'ultima riga costa $T(1) \cdot 4^{\log n} = n^2$. Sommando i costi per livello di tutte le righe, si ottiene:

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log n - 1} n^2/2^{2i} \cdot 4^i + n^2 \\ &= n^2 \sum_{i=0}^{\log n - 1} \frac{2^{2i}}{2^{2i}} + n^2 \\ &= n^2 \sum_{i=0}^{\log n - 1} 1 + n^2 \\ &= n^2 \log n + n^2 \end{aligned}$$

da cui si ottiene:

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

8.2 Metodo della sostituzione

L'idea è quella di ipotizzare una soluzione, basata sulla propria esperienza, e dimostrare che questa soluzione è vera tramite induzione.

8.2.1 Esempio semplice, facile da dimostrare

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Guess: $T(n) = O(n)$; dobbiamo dimostrare che $\exists c > 0, m \geq 0 : T(n) \leq cn, \forall n \geq m$.

- **Caso base**: $T(1) = 1 \leq 1 \cdot c, \forall c \geq 1$.
- **Ipotesi induttiva**: $\forall n' < n : T(n') \leq cn'$.
- **Caso induttivo**:

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + n \\ &\leq c\lfloor n/2 \rfloor + n && \text{Sostituzione dell'ipotesi induttiva} \\ &\leq cn/2 + n && \text{Intero inferiore} \\ &= (c/2 + 1)n \\ &\leq cn \end{aligned}$$

L'ultima disequazione è vera se e solo se

$$(c/2 + 1) \leq c \Leftrightarrow c \geq 2$$

Quindi abbiamo dimostrato, scegliendo $c = 2$, che $T(n) \leq 2n = O(n)$. Nota: abbiamo anche ottenuto una stima precisa delle costanti coinvolte. E' ragionevole?

$$n + n/2 + n/4 + n/8 + \ldots + 1$$

8.2.2 Cosa succede se si sbaglia l'intuizione

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

E' facile vedere che $T(n) = \Theta(n^2)$. Supponiamo invece di voler dimostrare che $T(n) = O(n)$. E' possibile? No.

- **Ipotesi induttiva**: Partendo dall'ipotesi induttiva $T(n') \leq cn'$, per qualunque $n' < n$, si ottiene:

$$\begin{aligned} T(N) &= n + T(n-1) \\ &\leq c(n-1) + n \\ &\leq (c+1)n - c \\ &\leq (c+1)n \\ &\not\leq cn \end{aligned}$$

L'ultima disequazione non è soddisfacibile a causa del termine di ordine superiore.

8.2.3 Problema: difficoltà matematica

Ci sono casi in cui è possibile ipotizzare la soluzione giusta, ma poi è difficile risolvere la parte induttiva.

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

Supponiamo la soluzione sia $O(n)$. Dobbiamo dimostrare che $T(n) \leq cn, c \geq 0$.

- **Ipotesi induttiva**:

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 \\ &\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1 \\ &\not\leq cn \end{aligned}$$

  Ovviamente l'ultima disequazione è falsa. Cosa fare? Il problema qui è che la soluzione è giusta, ma non riusciamo a dimostrarlo a causa di un fattore di ordine inferiore (1).

- **Ipotesi induttiva, più forte** Proviamo con un'ipotesi induttiva più forte: sottraiamo un termine di ordine inferiore:

$$T(n) \leq cn - b$$

  Proviamo ad utilizzarla:

$$\begin{aligned} T(n) &\leq (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b \end{aligned}$$

  L'ultima disequazione è vera se

$$-2b + 1 \leq -b \Leftrightarrow b \geq 1.$$

- **Caso base**:

$$T(1) = 1 \leq c - b$$

  che è vera per $\forall c \geq b + 1$.

8.2.4 Problema con i casi base

$$T(n) = \begin{cases} 2T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Guess: $T(n) = O(n \log n) \Leftrightarrow T(n) \leq cn \log n$.

- **Passo induttivo**:

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &\leq 2cn/2 \log n/2 + n \\ &= cn \log n/2 + n \\ &= cn(\log n - 1) + n \\ &= cn \log n - cn + n \\ &\leq cn \log n \end{aligned}$$

  L'ultima disequazione è vera per $c \geq 1$.

- Proviamo con il passo base:

$$T(n) = 1 \not\leq c \cdot 1 \log 1 = 0$$

E' falso, ma non è un problema. Non a caso si chiama notazione asintotica; il valore iniziale di m lo possiamo scegliere noi.

Quindi:

$$\begin{aligned} T(2) &= 2T(\lfloor 1 \rfloor) + 2 = 4 \leq c \cdot 2 \log 2 \\ T(3) &= 2T(\lfloor 1 \rfloor) + 3 = 5 \leq c \cdot 3 \log 3 \\ T(4) &= 2T(\lfloor 2 \rfloor) + 4 \end{aligned}$$

E' facile trovare un valore di c che soddisfi le prime due disequazioni; per la terza, non viene espressa come $T(1)$, quindi non è necessario andare avanti.

8.2.5 All together now!

$$T(n) = \begin{cases} 9T(\lfloor n/3 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases}$$

- **Ipotesi induttiva**: $T(n) = O(n^2), T(n) \leq cn^2$
- **Passo induttivo**:

$$\begin{aligned} T(n) &= 9T(\lfloor n/3 \rfloor) + n \\ &\leq 9c(\lfloor n/3 \rfloor)^2 + n \\ &\leq 9c(n^2/9) + n \\ &= cn^2 + n \\ &\not\leq cn^2 \end{aligned}$$

L'ipotesi è corretta tuttavia sottraiamo un valore di ordine inferiore.

- **Ipotesi induttiva più forte**: $T(n) \leq c(n^2 - n)$
- **Passo induttivo**:

$$\begin{aligned} T(n) &= 9T(\lfloor n/3 \rfloor) + n \\ &\leq 9c(\lfloor n/3 \rfloor^2 - \lfloor n/3 \rfloor) + n \\ &\leq cn^2 - 3cn + n \\ &\leq cn^2 - cn \end{aligned}$$

L'ultima disequazione è vera se e solo se:

$$\begin{aligned} cn^2 - 3cn + n &\leq cn^2 - cn \\ &\Leftrightarrow \\ -3cn + n &\leq -cn \\ &\Leftrightarrow \\ 1 &\leq 2c \\ &\Leftrightarrow \\ c &\geq 1/2 \end{aligned}$$

- **Passo base**:
  - $T(1) = 1 \leq c(1^2 - 1) = 0$, falso
  - $T(2) = 9T(0) + 2 = 11 \leq c(4 - 2) \Rightarrow c \geq \frac{11}{2}$
  - $T(3) = 9T(1) + 3 = 12 \leq c(9 - 3) \Rightarrow c \geq \frac{12}{6}$
  - $T(4) = 9T(1) + 4 = 13 \leq c(16 - 4) \Rightarrow c \geq \frac{13}{12}$
  - $T(5) = 9T(1) + 5 = 14 \leq c(25 - 5) \Rightarrow c \geq \frac{14}{20}$
  - $T(6) = 9T(2) + 6 \ldots$

Si noti che da $T(6)$ in poi non è più necessario andare avanti a dimostrare casi base, in quanto $T(6)$ dipende da $T(2)$ che è già stato dimostrato; possiamo quindi utilizzare l'ipotesi induttiva e $T(6)$ rientra nel passo induttivo.
E' importante notare che questo esercizio è artificialmente complicato: se si sceglieva come ipotesi induttiva più stretta $T(n) \leq cn^2 - bn$, invece che $T(n) \leq c(n^2 - n)$, la difficoltà sui casi base non si sarebbe presentata.

9 Ricorrenze lineari con partizione bilanciata

Per algoritmi di tipo divide-et-impera in cui si partizionino i dati in modo bilanciato, il problema originario di dimensione n è diviso in a sottoproblemi di dimensione n/b ciascuno. Se si dividono i dati e/o si ricombinano i risultati in tempo polinomiale, la funzione di complessità $T(n)$ può essere espressa in termini di $aT(n/b) + cn^\beta$. I parametri a e b sono costanti intere tali che $a \geq 1$, perché è fatta almeno una chiamata, e $b \geq 2$, in quanto il problema è suddiviso in almeno due sottoproblemi, mentre c e β sono, come prima, costanti reali tali che $c > 0$ e $\beta \geq 0$.

Teorema 1 (Ricorrenze lineari con partizione bilanciata)**.** Siano a e b costanti intere tali che $a \geq 1$ e $b \geq 2$, e c, β costanti reali tali che $c > 0$ e $\beta \geq 0$. Sia $T(n)$ data dalla relazione di ricorrenza:

$$T(n) = \begin{cases} aT(n/b) + cn^\beta & n > 1 \\ \Theta(1) & n \leq 1 \end{cases}$$

Posto $\alpha = \log a / \log b = \log_b a$, allora:

$$T(n) = \begin{cases} O(n^\alpha) & \alpha > \beta \\ O(n^\alpha \log n) & \alpha = \beta \\ O(n^\beta) & \alpha < \beta \end{cases}$$

Dimostrazione. Si supponga dapprima per semplicità che n sia una potenza di b, cioè $n = b^k$ per un k intero. Sostituendo successivamente si ottiene:

Livello	**Dimensione**	**Costo chiamata**	**N. chiamate**	**Costo livello**
0	b^k	$cb^{k\beta}$	1	$cb^{k\beta}$
1	b^{k-1}	$cb^{(k-1)\beta}$	a	$acb^{(k-1)\beta}$
2	b^{k-2}	$cb^{(k-2)\beta}$	a^2	$a^2cb^{(k-2)\beta}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
i	b^{k-i}	$cb^{(k-i)\beta}$	a^i	$a^icb^{(k-i)\beta}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$k-1$	b	cb^β	a^{k-1}	$a^{k-1}cb^\beta$
k	1	d	a^k	da^k

Sommando i costi totali di tutti i livelli, si ottiene:

$$T(n) = da^k + cb^{k\beta} \sum_{i=0}^{k-1} \frac{a^i}{b^{i\beta}} = da^k + cb^{k\beta} \sum_{i=0}^{k-1} (\frac{a}{b^\beta})^i$$

dove il primo termine è dato dall'ultimo livello e nella sommatoria abbiamo fattorizzato il valore $cb^{k\beta}$.
Osservazioni:

- Si osservi che $a^k = a^{\log_b n} = a^{\log n / \log b} = 2^{\log a \log n / \log b} = n^{\log a / \log b} = n^\alpha$.
- Si osservi inoltre che $a = b^\alpha$, in quanto $\alpha = \log a / \log b \Rightarrow \alpha \log b = \log a \Rightarrow \log b^\alpha = \log a$.
- Ponendo $q = \frac{a}{b^\beta} = \frac{b^\alpha}{b^\beta} = b^{\alpha-\beta}$, la relazione può essere scritta come:

$$T(n) = dn^\alpha + cb^{k\beta} \sum_{i=0}^{k-1} q^i$$

Si individuano tre casi, a seconda che α sia maggiore, uguale, o minore di β, e che quindi q risulti, rispettivamente, maggiore, uguale, o minore di 1.

1. Se $\alpha > \beta$, allora $q = b^{\alpha-\beta} > 1$:

$$\begin{aligned}
T(n) &= dn^\alpha + cb^{k\beta} \textstyle\sum_{i=0}^{k-1} q^i \\
&= n^\alpha d + cb^{k\beta}[(q^k - 1)/(q - 1)] && \text{Serie geometrica finita} \\
&\leq n^\alpha d + cb^{k\beta} q^k/(q - 1) && \text{Disequazione} \\
&= n^\alpha d + \frac{cb^{k\beta} a^k}{b^{k\beta}}/(q - 1) && \text{Sostituzione } q \\
&= n^\alpha d + ca^k/(q - 1) && \text{Passi algebrici} \\
&= n^\alpha [d + c/(q - 1)] && a^k = n^\alpha, \text{raccolta termini}
\end{aligned}$$

   e quindi $T(n)$ è $O(n^\alpha)$;

2. Se $\alpha = \beta$, allora $q = b^{\alpha-\beta} = 1$

$$\begin{aligned}
T(n) &= dn^\alpha + cb^{k\beta} \textstyle\sum_{i=0}^{k-1} q^i \\
&= n^\alpha d + cn^\beta k && q^i = 1^i = 1 \\
&= n^\alpha d + cn^\alpha k && \alpha = \beta \\
&= n^\alpha (d + ck) && \text{Raccolta termini} \\
&= n^\alpha [d + c \log n / \log b] && k = \log_b n
\end{aligned}$$

   e pertanto $T(n)$ è $O(n^\alpha \log n)$;

3. Se $\alpha < \beta$, allora $q = b^{\alpha-\beta} < 1$:

$$\begin{aligned}
T(n) &= dn^\alpha + cb^{k\beta} \textstyle\sum_{i=0}^{k-1} q^i \\
&= n^\alpha d + cb^{k\beta}[(q^k - 1)/(q - 1)] && \text{Serie geometrica finita} \\
&= n^\alpha d + cb^{k\beta}[(1 - q^k)/(1 - q)] && \text{Inversione} \\
&\leq n^\alpha d + cb^{k\beta}[1/(1 - q)] && \text{Disequazione} \\
&= n^\alpha d + cn^\beta/(1 - q) && b^k = n
\end{aligned}$$

   e quindi $T(n)$ è $O(n^\beta)$.

Nel caso che n non sia una potenza di b, si può considerare un problema di dimensione più grande, p.e. n', tale che n' sia la più piccola potenza di b maggiore di n. Poiché $n' \leq bn$ e b è una costante, la dimensione aumenta solo di un fattore costante, e la soluzione rimane valida in ordine di grandezza. Inoltre, poiché si è interessati all'analisi del caso pessimo, la soluzione della relazione di ricorrenza si applica anche al caso in cui valga il "$\leq$" anziché l'"=". Infine, la soluzione resta valida anche se $T(n)$ è uguale ad una costante per $n \leq h$, con h costante intera non necessariamente uguale ad 1, oppure se al posto di n/b si ha $\lfloor n/b \rfloor$, oppure $\lceil n/b \rceil$. □

9.1 Ricorrenze lineari con partizione bilanciata, versione estesa

Il teorema che abbiamo visto e dimostrato che si occupa di ricorrenze lineari con partizione bilanciata è una versione "semplificata" di un teorema più generale. La versione semplificata è più facilmente dimostrabile e più facile da applicare in alcuni casi. La versione generale non viene dimostrata qui e richiede più attenzione; il vantaggio è che riesce a gestire una classe più ampia di funzioni.

Teorema 2. Sia $a \geq 1$, $b > 1$, $f(n)$ asintoticamente positiva, e sia

$$T(n) = \begin{cases} aT(n/b) + f(n) & n > 1 \\ \Theta(1) & n \leq 1 \end{cases}$$

Allora $T(n)$ può essere limitata asintoticamente in uno dei modi seguenti:

- se $f(n) = O(n^{\log_b a - \epsilon})$ per qualche $\epsilon > 0$, allora
$$T(n) = \Theta(n^{\log_b a})$$
- se $f(n) = \Theta(n^{\log_b a})$, allora
$$T(n) = \Theta(n^{\log_b a} \log n) = \Theta(f(n) \log n)$$
- se $f(n) = \Omega(n^{\log_b a + \epsilon})$ per qualche $\epsilon > 0$, e $af(n/b) \leq cf(n)$ per qualche costante $c < 1$ e per ogni n sufficientemente grande, allora
$$T(n) = \Theta(f(n))$$

A parole, cosa ci dice questo teorema? Se si confronta $n^{\log_b a}$, si ottiene:

- se $n^{\log_b a}$ è più grande di $f(n)$ di un fattore polinomiale n^ϵ, allora $T(n)$ è dominato da $n^{\log_b a}$;
- se $n^{\log_b a}$ è $\Theta(n^{\log_b a})$, allora nessuno dei due domina, e viene aggiunto un fattore logaritmico;
- se $n^{\log_b a}$ è più piccolo di $f(n)$ di un fattore polinomiale n^ϵ, allora $T(n)$ è dominato da $f(n)$.

9.2 Ricorrenze lineari di ordine costante

Teorema 3 (Ricorrenze lineari di ordine costante)**.** Siano $a_1, a_2, \ldots, a_h$ costanti intere non negative, con h costante positiva, c e β costanti reali tali che $c > 0$ e $\beta \geq 0$, e sia $T(n)$ definita dalla relazione di ricorrenza:

$$T(n) = \begin{cases} \sum_{1 \leq i \leq h} a_i T(n-i) + cn^\beta & n > m \\ \Theta(1) & n \leq m \leq h \end{cases}$$

Posto $a = \sum_{1 \leq i \leq h} a_i$, allora:

1. $T(n)$ è $\Theta(n^{\beta+1})$, se $a = 1$,
2. $T(n)$ è $\Theta(a^n n^\beta)$, se $a \geq 2$.

Dimostrazione. La dimostrazione si trova nel libro, per coloro che sono interessati. □

9.3 Esempi di applicazione

	Ricorrenza	**a**	**b**	$\log_b a$	**Caso**	**Funzione**
(1)	$T(n) = 9T(n/3) + n$	9	3	2	(1)	$T(n) = \Theta(n^2)$
(2)	$T(n) = T(2n/3) + 1$	1	$\frac{3}{2}$	0	(2)	$T(n) = \Theta(\log n)$
(3)	$T(n) = 3T(n/4) + n \log n$	3	4	≈ 0.79	(3)	$T(n) = \Theta(n \log n)$
(4)	$T(n) = 2T(n/2) + n \log n$	2	2	1	-.	Non applicabile

	Ricorrenza	**a**	β	**Caso**	**Funzione**
(5)	$T(n) = T(n-10) + n^2$	1	2	(1)	$T(n) = \Theta(n^3)$
(6)	$T(n) = T(n-2) + T(n-1) + 1$	1	0	(2)	$T(n) = 2^n$

1. $f(n) = n = O(n^{\log_b a - \epsilon}) = O(n^{2-\epsilon})$, con $\epsilon > 1$
2. $f(n) = n^0 = \Theta(n^{\log_b a}) = \Theta(n^0)$
3. $f(n) = n \log n = \Omega(n^{\log_4 3 + \epsilon})$, con $\epsilon < 1 - \log_4 3 \approx 0.208$

   Dobbiamo anche dimostrare che: $\exists c \leq 1, m \geq: af(n/b) \leq cf(n), \forall n \geq m$

   $$\begin{aligned} af(n/b) &= 3n/4 \log n/4 \\ &= 3/4n \log n - 3/4n \log 2 \\ &\leq 3/4n \log n \\ &\overset{?}{\leq} cn \log n \end{aligned}$$

   L'ultima è diseguazione è soddisfatta da $c = 3/4$ e $m = 1$.
4. $f(n) = n \log n \neq O(n^{1-\epsilon})$, con $\epsilon > 0$
   $f(n) = n \log n \neq \Theta(n)$
   $f(n) = n \log n \neq \Omega(n^{1+\epsilon})$, con $\epsilon > 0$
   Nessuno dei tre casi è applicabile e bisogna utilizzare altri metodi.
5. Poiché $a = 1$, il costo è polinomiale.
6. Poiché $a = 2$, il costo è esponenziale.