Capitolo 2

Metodologie di analisi

2.1 La notazione asintotica

Definizione 2.1. Data una funzione $f(n)$, definiamo:

- $O(f(n)) = \{g(n) : \exists c > 0 \wedge n_0 \geq 0 : g(n) \leq cf(n), \forall n \geq n_0\}$ ($g(n)$ cresce *al più* come $f(n)$);
- $\Omega(f(n)) = \{g(n) : \exists c > 0 \wedge n_0 \geq 0 : g(n) \geq cf(n), \forall n \geq n_0\}$ ($g(n)$ cresce *almeno* come $f(n)$);
- $\Theta(f(n)) = \{g(n) : \exists c_1, c_2 > 0 \wedge n_0 \geq 0 : c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}$ ($g(n)$ cresce *esattamente* come $f(n)$).

Proprietà 2.1. Date due funzioni $f(n)$ e $g(n)$, risulta $g(n) = \Theta(f(n))$ se e solo se $g(n) = O(f(n))$ e $g(n) = \Omega(f(n))$.

Proprietà 2.2. Θ gode della proprietà simmetrica: $g(n) = \Theta(f(n))$ se e solo se $f(n) = \Theta(g(n))$.

Proprietà 2.3. O, Ω sono simmetriche trasposte: $f(n) = \Omega(g(n))$ se e solo se $g(n) = O(f(n))$.

Proprietà 2.4. Per tutte e tre le notazioni vale la proprietà transitiva: per la notazione O, ad esempio, $f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$.

Teorema 2.1. *Siano $f(n)$ e $g(n)$ funzioni positive:*

1. *Se $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = +\infty$ allora $f(n) = \Omega(g(n))$ o, equivalentemente, $g(n) = O(f(n))$.*
2. *Se $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = 0$ allora $f(n) = O(g(n))$ o, equivalentemente, $g(n) = \Omega(f(n))$.*
3. *Se $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = l \neq 0$ allora $f(n) = \Theta(g(n))$ o, equivalentemente, $g(n) = \Theta(f(n))$.*

Dimostrazione. Vediamo i vari casi.

1. Sia $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = +\infty$, allora esistono $M > 0$ e $n_0 > 0$ tali che $\frac{f(n)}{g(n)} \geq M$ per ogni $n \geq n_0$. Quindi $f(n) \geq M \cdot g(n)$ per ogni $n \geq n_0$. Questo significa che $f(n) = \Omega(g(n))$, ovvero $g(n) = O(f(n))$.
2. Sia $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = 0$, allora preso $\epsilon > 0$ esiste $n_0 > 0$ tale che $\frac{f(n)}{g(n)} \leq \epsilon$ per ogni $n \geq n_0$. Quindi $f(n) \leq \epsilon \cdot g(n)$ per ogni $n \geq n_0$. Questo significa che $f(n) = O(g(n))$, ovvero $g(n) = \Omega(f(n))$.
3. Sia $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = l \neq 0$, allora preso $\epsilon > 0$ esiste $n_0 > 0$ tale che $\left|\frac{f(n)}{g(n)} - l\right| \leq \epsilon$ per ogni $n \geq n_0$. Quindi $\epsilon \leq \frac{f(n)}{g(n)} - l \leq \epsilon$, ovvero $l - \epsilon \leq \frac{f(n)}{g(n)} \leq l + \epsilon$. Di conseguenza $(l-\epsilon)\cdot g(n) \leq f(n) \leq (l+\epsilon)\cdot g(n)$. Questo significa che $f(n) = \Theta(g(n))$ ovvero $g(n) = \Theta(f(n))$.

□

2.2 Delimitazioni inferiori e superiori

Definiamo il costo di esecuzione di un algoritmo e la complessità di un problema.

Definizione 2.2. Un algoritmo $\mathcal{A}$ ha costo di esecuzione $O(f(n))$ su istanze di ingresso di dimensione n e rispetto ad una certa risorsa di calcolo, se la quantità r di risorsa *sufficiente* per eseguire $\mathcal{A}$ su una qualunque istanza di dimensione n verifica la relazione $r(n) = O(f(n))$.

Definizione 2.3. Un problema $\mathcal{P}$ ha complessità $O(f(n))$ rispetto ad una data risorsa di calcolo se *esiste* un algoritmo che risolve $\mathcal{P}$ il cui costo di esecuzione rispetto a quella risorsa è $O(f(n))$.

Definizione 2.4. Un algoritmo $\mathcal{A}$ ha costo di esecuzione $\Omega(f(n))$ su istanze di dimensione n e rispetto ad una certa risorsa di calcolo, se la massima quantità r di risorsa *necessaria* per eseguire $\mathcal{A}$ su istanze di dimensione n verifica la relazione $r(n) = \Omega(f(n))$.

Definizione 2.5. Un problema $\mathcal{P}$ ha complessità $\Omega(f(n))$ rispetto ad una data risorsa di calcolo se *ogni* algoritmo che risolve $\mathcal{P}$ ha costo di esecuzione $\Omega(f(n))$ rispetto a quella risorsa.

Definizione 2.6. Dato un problema $\mathcal{P}$ con complessità $\Omega(f(n))$ rispetto ad una data risorsa di calcolo, un algoritmo che risolve $\mathcal{P}$ è *ottimo* se ha costo di esecuzione $O(f(n))$ rispetto a quella risorsa.

2.3 Metodi di analisi

Per analizzare il tempo di esecuzione di un algoritmo, solitamente si usa distinguere fra tre diverse categorie di istanze, a parità di dimensione; sia $T(\mathcal{I})$ il tempo di esecuzione dell'algoritmo sull'istanza $\mathcal{I}$:

Caso peggiore:

$$T_{worst}(n) = \max_{\substack{\mathcal{I}\text{: istanze di} \\ \text{dimensione } n}} T(\mathcal{I})$$

Caso migliore:

$$T_{best}(n) = \min_{\substack{\mathcal{I}\text{:istanze di} \\ \text{dimensione } n}} T(\mathcal{I})$$

Caso medio: sia $P(\mathcal{I})$ la probabilità di occorrere dell'istanza $\mathcal{I}$:

$$T_{avg}(n) = \sum_{\substack{\mathcal{I}\text{:istanze di} \\ \text{dimensione } n}} (T(\mathcal{I}) \cdot P(\mathcal{I}))$$

2.4 Analisi di algoritmi ricorsivi

2.4.1 Metodo di iterazione

L'idea è quella di ridurre la ricorsione ad una sommatoria dipendente solo dalla dimensione del problema iniziale.

Esempio 2.1. Sia data la seguente relazione di ricorrenza e assumiamo, per semplicità di analisi, che n sia una potenza di 3:

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 9 \cdot T(n/3) + n & \text{se } n > 1 \end{cases}$$

Svolgendo la ricorsione otteniamo:

$$\begin{aligned} T(n) &= 9 \cdot T(n/3) + n \\ &= 9 \cdot (9 \cdot T(n/9) + n/3) + n \\ &= 9^2 \cdot T(n/3^2) + 9 \cdot n/3 + n \\ &= ... \\ &= 9^i \cdot T(n/3^i) + \sum_{j=0}^{i-1} (9/3)^j n \end{aligned}$$

Dal momento che $n/3^i = 1$ quando $i = \log_3 n$, risulta:

$$T(n) = 9^{log_3 n} + n \cdot \sum_{j=0}^{\log_3 n - 1} 3^j$$

Poiché $\log_3 n = \log_9 n \cdot \log_3 9$, risulta $9^{\log_3 n} = 9^{\log_9 n \cdot \log_3 9} = n^2$; usando la serie geometrica, si ottiene:

$$T(n) = n^2 + \frac{3^{\log_3 n} - 1}{3 - 1} = n^2 + \frac{n-1}{2} = \Theta(n^2)$$

2.4.2 Metodo di sostituzione

L'idea è di intuire la soluzione della relazione di ricorrenza ed utilizzare il principio di induzione per dimostrare che l'intuizione è corretta.

Esempio 2.2. Sia data la seguente relazione di ricorrenza:

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

Intuizione: $T(n) = O(n)$, ossia dimostrare che esistono $c > 0$ e $n_0 > 0$ tali che $T(n) \leq c \cdot n, \forall n \geq n_0$

Passo base: proviamo $n = 1$
$T(1) = 1 \leq c \cdot 1 \rightarrow c \geq 1$

Ipotesi induttiva: prendiamo $n > 1$ e supponiamo di avere $c > 0$ tale per cui $T(\lfloor n/2 \rfloor) \leq c \cdot \lfloor n/2 \rfloor$.

Passo induttivo: sia $n > 1$
$T(n) = T(\lfloor n/2 \rfloor) + n$
$\leq c \cdot \lfloor n/2 \rfloor + n$ (per ipotesi induttiva)
$\leq c \cdot \frac{n}{2} + n$
Rimane da provare che $c \cdot \frac{n}{2} + n \leq c \cdot n$; ciò risulta essere vero per $c \geq 2$.

Affinché siano verificati sia il caso base che il passo induttivo, occorre prendere un qualsiasi $c \geq 2$.

Esempio 2.3. Per illustrare altre sottigliezze relative all'uso del metodo di sostituzione, consideriamo la seguente relazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n = 1, 2 \\ 9 \cdot T(\lfloor n/3 \rfloor) + n & \text{se } n > 2 \end{cases}$$

Intuizione: $T(n) = O(n^2)$, ossia cerchiamo di dimostrare che esistono $c > 0$ e $n_0 > 0$ tali che $T(n) \leq c \cdot n^2, \forall n \geq n_0$

Passo base: $T(1) = T(2) = 1 \leq c \cdot 1^2 \leq c \cdot 2^2 \rightarrow c \geq 1$

Ipotesi induttiva: assumiamo che la disuguaglianza sia soddisfatta per ogni $k < n$

Passo induttivo: sia $n > 2$
$T(n) = 9 \cdot T(\lfloor n/3 \rfloor) + n$
$\leq 9 \cdot c \cdot \lfloor n/3 \rfloor^2 + n$ (per ipotesi induttiva)
$\leq 9 \cdot c \cdot (n/3)^2 + n$
$= c \cdot n^2 + n$

Ora dovremmo provare che $c \cdot n^2 + n \leq c \cdot n^2$. Questo è evidentemente falso! Possiamo risolvere facilmente il problema usando un'ipotesi induttiva più forte, in modo da far comparire un addendo negativo negativo di ordine inferiore a n^2 che, sommato ad n, faccia tornare i conti. Proviamo con l'ipotesi $T(k) \leq c \cdot (k^2 - k)$ per ogni $k < n$:

$$\begin{aligned} T(n) &= 9 \cdot T(\lfloor n/3 \rfloor) + n \\ &\leq 9 \cdot c \cdot ((n/3)^2 - n/3) + n \quad \text{(per ipotesi induttiva)} \\ &= c \cdot n^2 - 3 \cdot c \cdot n + n \end{aligned}$$

Rimane da provare che $c \cdot n^2 - 3 \cdot c \cdot n + n \leq c \cdot (n^2 - n)$, ovvero che $n \leq 2cn$. Ciò risulta essere vero per $c \geq 1/2$.

Abbiamo ora un altro problema: il passo base non è più verificato. Infatti $T(1) = 1 > c \cdot (1^2 - 1) = 0$. Ciò può essere risolto modificando la base n_0, che prima avevamo scelto pari a 1. Poiché $T(2) = 1 \leq c \cdot (4 - 2)$ per $c \geq 1/2$, le cose potrebbero funzionare con $n_0 = 2$. Dobbiamo però dare un buon fondamento all'induzione. Per esempio, il caso $T(3)$ non può essere considerato induttivo: esso si rifà a $T(1)$ che non è più un caso base. Dobbiamo quindi considerare tutti i casi che si riconducono a $T(1)$ come casi base e verificare che, per tutti questi, la condizione $T(n) \leq c(n^2 - n)$ sia verificata. In questo caso dobbiamo definire come casi base, oltre a $T(2)$, anche $T(3), T(4), T(5)$. Per $n \geq 6$ ci si riconduce a uno di questi. Vediamoli tutti in dettaglio:

$$\begin{array}{lll} T(3) & = 9 \cdot T(1) + 3 = 12 \quad \leq c \cdot (3^2 - 3) & \text{per } c \geq 2 \\ T(4) & = 9 \cdot T(1) + 4 = 13 \quad \leq c \cdot (4^2 - 4) & \text{per } c \geq 13/12 \\ T(5) & = 9 \cdot T(1) + 5 = 14 \quad \leq c \cdot (5^2 - 5) & \text{per } c \geq 7/10 \end{array}$$

Possiamo scegliere $n_0 = 2$ e $c \geq 2$.

2.4.3 Il teorema fondamentale delle ricorrenze

Si tratta di un metodo per analizzare algoritmi basati sulla tecnica del *dividi et impera*, in cui:

- un problema di dimensione n viene diviso in a sottoproblemi di dimensione n/b;
- dividere in sottoproblemi e combinare le soluzioni richiede tempo[1] $f(n)$.

La relazione di ricorrenza corrispondente a questo scenario è la seguente:

$$T(n) = a \cdot T\Big(\frac{n}{b}\Big) + f(n) \tag{2.1}$$

Per analizzare la relazione di ricorrenza, consideriamo l'albero della ricorsione ed assumiamo che n sia una potenza esatta di b e che la ricorsione si fermi quando $n = 1$. Facciamo intanto alcune osservazioni. Vi invito a disegnare l'albero della ricorsione e di verificare le seguenti proprietà.

Proprietà 2.5. I sottoproblemi al livello i dell'albero della ricorsione hanno dimensione n/b^i.

Proprietà 2.6. Il contributo di un nodo di livello i al tempo di esecuzione (escluso il tempo speso nelle chiamate ricorsive) è $f(n/b^i)$.

Proprietà 2.7. Il numero di livelli nell'albero della ricorsione è $\log_b n$.

Proprietà 2.8. Il numero di nodi al livello i dell'albero della ricorsione è a^i.

Il costo totale è dato dalla somma dei costi locali dei nodi dell'albero. Sommiamo per livelli e usiamo le proprietà appena elencate. Si può riscrivere la relazione di ricorrenza (2.1) nella seguente forma:

$$T(n) = \sum_{i=0}^{\log_b n} a^i \cdot f\Big(\frac{n}{b^i}\Big) \tag{2.2}$$

La soluzione della (2.2) è data dal seguente teorema, noto come *Teorema Fondamentale delle Ricorrenze* o come *Master Theorem* (Cormen).

Teorema 2.2. *La relazione di ricorrenza*

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ a \cdot T(n/b) + f(n) & \text{se } n > 1 \end{cases}$$

ha soluzione:

[1] Visto che $f(n)$ corrisponde a un tempo di esecuzione possiamo assumere $f(n) \geq 0$ per ogni n.

1. $T(n) = \Theta(n^{\log_b a})$, *se* $f(n) = O(n^{\log_b a - \epsilon})$ *per* $\epsilon > 0$;
2. $T(n) = \Theta(n^{\log_b a} \cdot \log n)$, *se* $f(n) = \Theta(n^{\log_b a})$;
3. $T(n) = \Theta(f(n))$, *se* $f(n) = \Omega(n^{\log_b a + \epsilon})$ *per* $\epsilon > 0$ *e* $a \cdot f(n/b) \leq c \cdot f(n)$ *per* $c < 1$ *ed* n *sufficientemente grande.*

Dimostrazione. Assumiamo per semplicità che n sia una potenza esatta di b.

Caso 1: riscriviamo il termine generico della sommatoria 2.2:

$$a^i \cdot f\Big(\frac{n}{b^i}\Big) = O\Big(a^i \cdot \Big(\frac{n}{b^i}\Big)^{\log_b a - \epsilon}\Big) = O\Big(n^{\log_b a - \epsilon} \cdot \Big(\frac{a \cdot b^\epsilon}{b^{\log_b a}}\Big)^i\Big) = O(n^{\log_b a - \epsilon} \cdot (b^\epsilon)^i)$$

Limite superiore. Per limitare superiormente $T(n)$ si può scrivere:

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

Limite inferiore. Analizzando l'equazione 2.2 e considerando solo i tempi di esecuzione relativi ai nodi sull'ultimo livello dell'albero di ricorsione (con $i = \log_b n$), otteniamo[2]

$$T(n) \geq a^{\log_b n} = a^{\log_b a \cdot \log_a n} = n^{\log_b a}$$

e quindi $T(n) = \Omega(n^{\log_b a})$.

Dalle due limitazioni segue che $T(n) = \Theta(n^{\log_b a})$.

Caso 2: anche in questo caso, riscriviamo il termine generico della sommatoria:

$$a^i \cdot f\Big(\frac{n}{b^i}\Big) = \Theta\Big(a^i \cdot \Big(\frac{n}{b^i}\Big)^{\log_b a}\Big) = \Theta\Big(n^{\log_b a} \cdot \Big(\frac{a}{b^{\log_b a}}\Big)^i\Big) = \Theta(n^{\log_b a})$$

Da cui segue:

$$T(n) = \sum_{i=0}^{\log_b n} \Theta(n^{\log_b a}) = \Theta(n^{\log_b a} \cdot \log_b n).$$

Caso 3: come nel caso 1, vediamo separatamente i due limiti.

Limite inferiore. Sotto l'assunzione $a \cdot f(n/b) \leq c \cdot f(n)$ possiamo dedurre che

$$a^i\, f\Big(\frac{n}{b^i}\Big) = a^{i-1}\Big(a\, f\Big(\frac{n/b^{i-1}}{b}\Big)\Big) \leq a^{i-1}\Big(c\, f\Big(\frac{n}{b^{i-1}}\Big)\Big) = a^{i-2}\, c\,\Big(a\, f\Big(\frac{n/b^{i-2}}{b}\Big)\Big) \leq a^{i-2}\, c\Big(c\, f\Big(\frac{n}{b^{i-2}}\Big)\Big) \cdots$$

Iterando il ragionamento, otteniamo

$$a^i \cdot f(n/b^i) \leq c^i \cdot f(n)$$

Sfruttando l'equazione 2.2, la disuguaglianza appena ottenuta e le proprietà della serie geometrica con base $c < 1$, possiamo scrivere:

$$T(n) = \sum_{i=0}^{\log_b n} a^i \cdot f\Big(\frac{n}{b^i}\Big) \leq f(n) \cdot \sum_{i=0}^{+\infty} c^i = f(n) \cdot \frac{1}{1-c} = O(f(n))$$

Limite superiore. Nella relazione 2.2 consideriamo solo in termine che si ottiene per $i = 0$ e otteniamo $T(n) = \Omega(f(n))$

Dalle due limitazioni concludiamo $T(n) = \Theta(f(n))$. □

[2] Vi ricordo la regola di cambiamento di base per i logaritmi: $\log_b n = \log_b a \cdot \log_a n$

Esempio 2.4. Consideriamo la seguente relazione di ricorrenza:

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ T(n/2) + O(1) & \text{altrimenti} \end{cases}$$

Si ha $f(n) = O(1) = \Theta(n^{\log_2 1})$. Ci troviamo, dunque, nel caso 2 del teorema fondamentale, e quindi risulta $T(n) = \Theta(\log n)$.

Esempio 2.5. Consideriamo la seguente relazione di ricorrenza:

$$T(n) = \begin{cases} O(1) & \text{se } n = 1, 2 \\ 9 \cdot T(n/3) + n & \text{altrimenti} \end{cases}$$

Si ha $f(n) = n = O(n^{\log_3 9 - \epsilon})$, basta per esempio prendere $\epsilon = 1/2$. Possiamo applicare il caso 1 del teorema fondamentale, e risulta $T(n) = \Theta(n^2)$.

Esempio 2.6. Consideriamo la seguente relazione di ricorrenza:

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ T(n/2) + n + \log_n + 1 & \end{cases}$$

Si ha $f(n) = n + \log n + 1 = \Theta(n) = \Omega(n^{\log_2 1 + \epsilon})$, , basta per esempio prendere ancora $\epsilon = 1/2$. L'equazione di ricorrenza *potrebbe* ricadere nel caso 3 del teorema fondamentale. Dobbiamo ancora trovare $c < 1$ tale che, per n sufficientemente grande, valga:

$$\begin{aligned} & a \cdot f(n/b) \leq c \cdot f(n) \\ & \to 1 \cdot \left(\frac{n}{2} + \log \frac{n}{2} + 1 \right) \leq c \cdot (n + \log n + 1) \\ & \to c \geq \frac{\frac{n}{2} + \log \frac{n}{2} + 1}{n + \log n + 1} \left(= \frac{1}{2} < 1 \text{ per } n \to +\infty \right) \end{aligned}$$

Poiché $\lim_{n \to +\infty} \frac{\frac{n}{2} + \log \frac{n}{2} + 1}{n + \log n + 1} = \frac{1}{2}$ possiamo affermare che esiste n_0 tale che per ogni $n \geq n_0$ si ha

$$\frac{3}{4} \geq \frac{\frac{n}{2} + \log \frac{n}{2} + 1}{n + \log n + 1}$$

È sufficiente prendere $c = 3/4$ affinché la disuguaglianza del caso 3 del teorema fondamentale sia verificata per n sufficientemente grande. Concludiamo $T(n) = \Theta(n)$.

Esempio 2.7. Consideriamo la seguente relazione di ricorrenza:

$$T(n) = \begin{cases} O(1) & \text{se } n = 1, 2 \\ 3 \cdot T(n/3) + n \cdot \log n & \text{altrimenti} \end{cases}$$

Nessuno dei casi del teorema principale può essere applicato; si potrebbe pensare di utilizzare il terzo caso, ma questo non è possibile poiché $n \cdot \log n$ è solo logaritmicamente, e non polinomialmente, più grande di $n^{\log_3 3} = n$.

[Esercizio] Studiate questa relazione di ricorrenza analizzando l'albero della ricorsione associato. (Suggerimento: seguite la traccia dell'Esempio 2.8.)

2.4.4 Analisi dell'albero della ricorsione

Studiamo l'albero delle chiamate ricorsive. Indichiamo le dimensioni dei problemi di ogni chiamata ricorsiva ed analizziamo la dimensione totale dei problemi ad ogni livello dell'albero.

Esempio 2.8. Consideriamo la seguente relazione di ricorrenza:

$$T(n) = \begin{cases} 1 & \text{se } n = 1, 2 \\ 9 \cdot T(n/3) + n^2 \cdot \log n & \text{altrimenti} \end{cases}$$

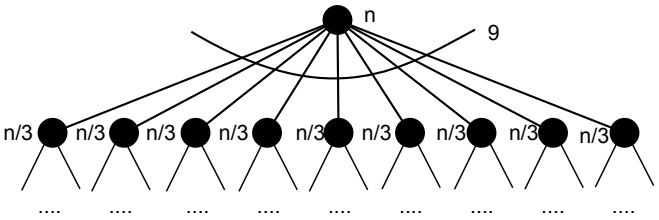

Figura 2.1: Parte iniziale dell'albero di ricorsione

Osserviamo intanto che i nodi a livello i di questo albero sono esattamente 9^i; infatti ogni nodo interno ha esattamente 9 figli. Calcoliamo il costo di ciascun livello dell'albero di ricorsione:

Liv. 0: rappresenta il problema di dimensione n. Ha costo locale $n^2 \cdot \log n \ \rightarrow$ totale $= n^2 \cdot \log n$.

Liv. 1: rappresenta sotto-problemi di dimensione $\frac{n}{3}$. Ciascuno ha costo locale $\left(\frac{n}{3}\right)^2 \cdot \log \frac{n}{3}$. Il costo totale è $9 \cdot \left(\frac{n}{3}\right)^2 \cdot \log \frac{n}{3}$.

Liv. 2: rappresenta sotto-problemi di dimensione $\frac{n}{3^2}$. Ciascuno ha costo locale $\left(\frac{n}{3^2}\right)^2 \cdot \log \frac{n}{3^2}$. Il costo totale è $9^2 \cdot \left(\frac{n}{3^2}\right)^2 \cdot \log \frac{n}{3^2}$.

Liv. i: rappresenta sotto-problemi di dimensione $\frac{n}{3^i}$. Ciascuno ha costo locale $\left(\frac{n}{3^i}\right)^2 \cdot \log \frac{n}{3^i}$. Il costo totale è $9^i \cdot \left(\frac{n}{3^i}\right)^2 \cdot \log \frac{n}{3^i} = n^2 \cdot \log n - i \cdot n^2 \cdot \log 3)$;

Foglie: rappresentano sotto-problemi di dimensione 1. Il loro livello k è tale che $\frac{n}{3^k} = 1$, quindi $k = \log_3 n$. In totale abbiamo $9^k = 9^{\log_3 n} = n^2$ foglie. Il costo totale è n^2

$$
\begin{aligned}
T(n) &= \underbrace{\sum_{i=0}^{\log_3 n - 1} (n^2 \cdot \log n - n^2 \cdot i \cdot \log 3)}_{\text{contributo nodi interni}} + \underbrace{9^{\log_3 n}}_{\text{contributo foglie}} \\
&= n^2 \cdot \log n \cdot \sum_{i=0}^{\log_3 n - 1} 1 - n^2 \cdot \log 3 \cdot \sum_{i=0}^{\log_3 n - 1} i + n^2 \\
&= n^2 \cdot \log n \cdot \log_3 n - n^2 \cdot \log 3 \cdot \frac{(\log_3 n - 1) \cdot (\log_3 n)}{2} + n^2 \\
&= n^2 \cdot \log n \cdot \log_3 n - n^2 \cdot \log 3 \cdot \frac{\log_3^2 n}{2} + n^2 \cdot \log 3 \cdot \frac{\log_3 n}{2} + n^2
\end{aligned}
$$

Risulta quindi $T(n) = \Theta(n^2 \cdot \log^2 n)$.

2.4.5 Cambiamenti di variabile

Si tratta di una tecnica che viene utilizzata quando la relazione di ricorrenza presenta delle radici. Vediamola con un esempio.

Esempio 2.9. Consideriamo la seguente relazione di ricorrenza:

$$
T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ T(\sqrt{n}) + O(1) & \text{altrimenti} \end{cases}
$$

Eseguiamo la sostituzione $n = 2^x$ (ossia $x = \log n$), da cui $\sqrt{n} = 2^{x/2}$ e $T(2^x) = T(2^{x/2}) + O(1)$; poniamo inoltre $T(2^x) = R(x)$, da cui otteniamo la relazione di ricorrenza $R(x) = R(x/2) + O(1)$, risolvibile con il teorema principale ($R(x) = O(\log x)$). Combinando le uguaglianze $T(2^x) = O(\log x)$ e $n = 2^x$, si ha $T(n) = O(\log \log n)$.