Corso di laurea in Informatica
Introduzione agli Algoritmi

Equazioni di ricorrenza

Angelo Monti

SAPIENZA
UNIVERSITÀ DI ROMA

Equazioni di ricorrenza

- 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**.

Esempio: equazione di ricorrenza della ricerca sequenziale ricorsiva in una lista A di n elementi (che restituisce *None* se l'elemento cercato non è presente oppure la sua posizione in caso contrario).

```
def RicercaR(x, A, i=0):
    if len(A)==i:
        return None
    if A[i] == x:
        return i
    return RicercaR(x, A, i+1)
```

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 0 \\ T(n-1) + \Theta(1) & \text{altrimenti} \end{cases}$$

Equazione di ricorrenza:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 0 \\ T(n-1) + \Theta(1) & \text{altrimenti} \end{cases}$$

Un'equazione di ricorrenza che definisce T(n) deve sempre includere almeno due termini: uno che rappresenta la parte ricorsiva (nell'esempio T(n-1)) e uno che esprime il costo computazionale delle operazioni eseguite al di fuori della chiamata ricorsiva.
Inoltre, deve sempre essere previsto un caso base.

Esploreremo quattro metodi principali per risolvere equazioni di ricorrenza:

- **Metodo iterativo**: consiste nell'espandere iterativamente la ricorrenza fino a identificare un pattern generale e ottenere la soluzione.
- **Metodo dell'albero**: si basa sulla rappresentazione della ricorrenza tramite un albero di calcolo, utile per visualizzare la struttura del problema e determinare il costo complessivo.
- **Metodo di sostituzione**: prevede di proporre una soluzione ipotetica e verificarne la validità tramite sostituzione nella ricorrenza.
- **Teorema principale**: offre un approccio sistematico per risolvere molte ricorrenze comuni, basandosi su una classificazione del termine di ricorrenza.

Esempio 1

Consideriamo la ricorrenza:

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

Espandiamo la ricorrenza iterativamente:

$$\begin{aligned} T(n) &= T(n-1) + \Theta(1) \\ &= \big(T(n-2) + \Theta(1)\big) + \Theta(1) \\ &= \big(T(n-3) + \Theta(1)\big) + \Theta(1) + \Theta(1) \end{aligned}$$

Proseguendo per k passi, otteniamo:

$$T(n) = T(n-k) + \sum_{i=1}^{k} \Theta(1).$$

Per $k = n-1$, raggiungiamo la condizione iniziale $T(1) = \Theta(1)$:

$$T(n) = T(1) + \sum_{i=1}^{n-1} \Theta(1).$$

Seguito Esempio 1:

$$T(n) = T(1) + \sum_{i=1}^{n-1} \Theta(1).$$

Poiché $T(1) = \Theta(1)$, possiamo scrivere:

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

Semplificando:

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

In termini asintotici, otteniamo:

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

Esempio 2

Consideriamo la seguente ricorrenza:

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

Espandiamo la ricorrenza iterativamente:

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + \Theta(1) \\ &= 2\left(2T\left(\frac{n}{4}\right) + \Theta(1)\right) + \Theta(1) \\ &= 2^2 T\left(\frac{n}{4}\right) + 2\Theta(1) + \Theta(1) \\ &= 2^2\left(2T\left(\frac{n}{8}\right) + \Theta(1)\right) + +2\Theta(1) + \Theta(1) \\ &= 2^3 T\left(\frac{n}{8}\right) + 2^2\Theta(1) + 2\Theta(1) + \Theta(1) \end{aligned}$$

Proseguendo per k passi, otteniamo:

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

Seguito Esempio 2:

Proseguendo per k passi, otteniamo:

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

La ricorsione termina quando $\frac{n}{2^k} = 1$ cioè quando $k = \log_2 n$. Nel caso base, $T(1) = \Theta(1)$, quindi:

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

- **Primo termine**: $2^{\log_2 n} = n$, quindi il termine dà $n \cdot \Theta(1) = \Theta(n)$
- **Secondo termine**: $\sum_{i=0}^{\log_2 n - 1} 2^i \Theta(1)$ la somma geometrica ha valore:

$$\sum_{i=0}^{\log_2 n - 1} 2^i = 2^{\log_2 n} - 1 = n - 1$$

quindi il termine dà: $(n-1) \cdot \Theta(1) = \Theta(n)$.

Sommando i contributi dei due termini otteniamo:

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

Metodo dell'Albero

Il metodo dell'albero è una tecnica grafica e intuitiva per analizzare ricorrenze, utile per comprendere come i costi si distribuiscono attraverso i livelli della ricorsione. Questa tecnica è particolarmente adatta per ricorrenze della forma:

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

dove $a > 1$, $b > 1$, e $f(n)$ rappresenta il contributo non ricorsivo.

Costruzione dell'Albero Ricorsivo

Per applicare il metodo dell'albero:

1. Espandiamo la ricorrenza iterativamente, suddividendo $T(n)$ nei suoi sottoproblemi fino a raggiungere la dimensione di base.
2. Sommiamo i contributi di ogni livello dell'albero.
3. Determiniamo il numero totale di livelli dell'albero e valutiamo il costo complessivo sommando i costi di tutti i livelli.

Esempio 1

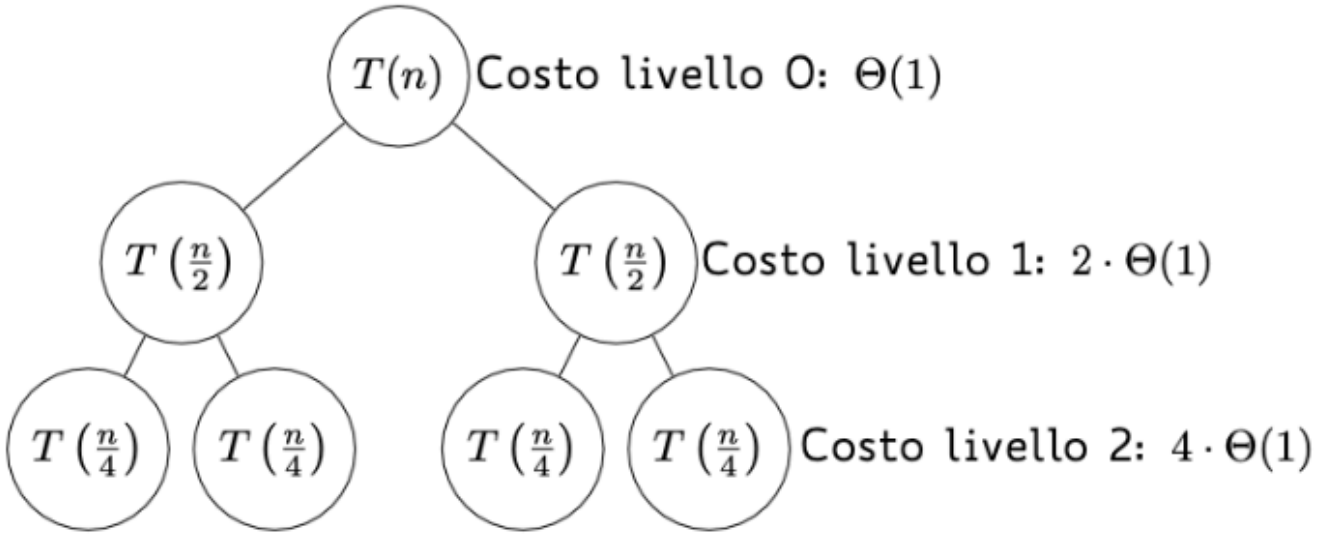
Consideriamo la ricorrenza:

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

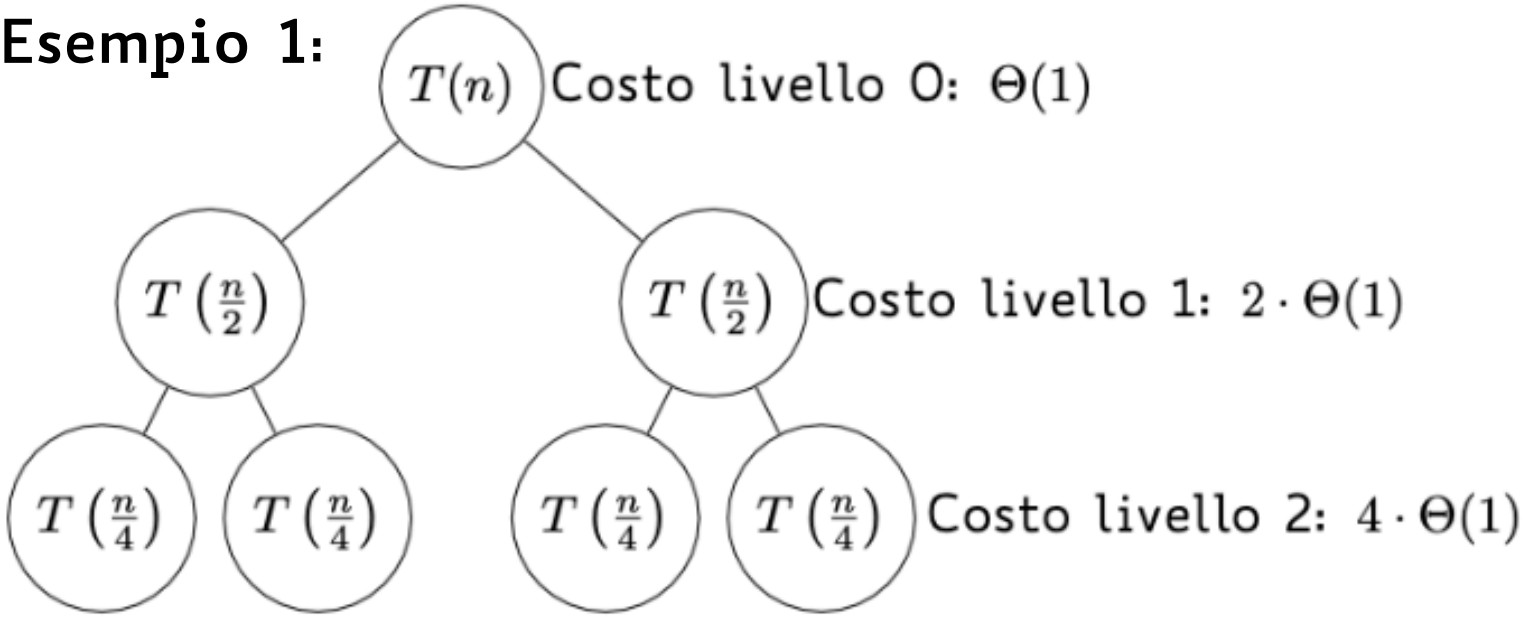
Costruzione dell'albero. - Al livello 0 (radice), il costo è $f(n) = \Theta(1)$. - Al livello 1, abbiamo 2 sottoproblemi di dimensione $\frac{n}{2}$, ognuno con costo $\Theta(1)$, per un costo totale di $2 \cdot \Theta(1)$. - Al livello k, ci sono 2^k sottoproblemi di dimensione $\frac{n}{2^k}$, ciascuno con costo $\Theta(1)$, per un costo totale di:

$$\text{Costo livello } k = 2^k \cdot \Theta(1).$$

Nell'iimmagine che segue c'i sono i primi livelli dell' albero di ricorsione per $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1)$:



Seguito Esempio 1:



Calcolo del numero di livelli. L'albero si espande fino a quando i sottoproblemi raggiungono dimensione 1, cioè:

$$\frac{n}{2^k} = 1 \implies k = \log_2 n.$$

Costo complessivo. Il costo totale è la somma dei costi su tutti i livelli:

$$\text{Costo totale} = \sum_{k=0}^{\log_2 n} 2^k \cdot \Theta(1) = \Theta(n).$$

Esempio 2

Consideriamo la ricorrenza:

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

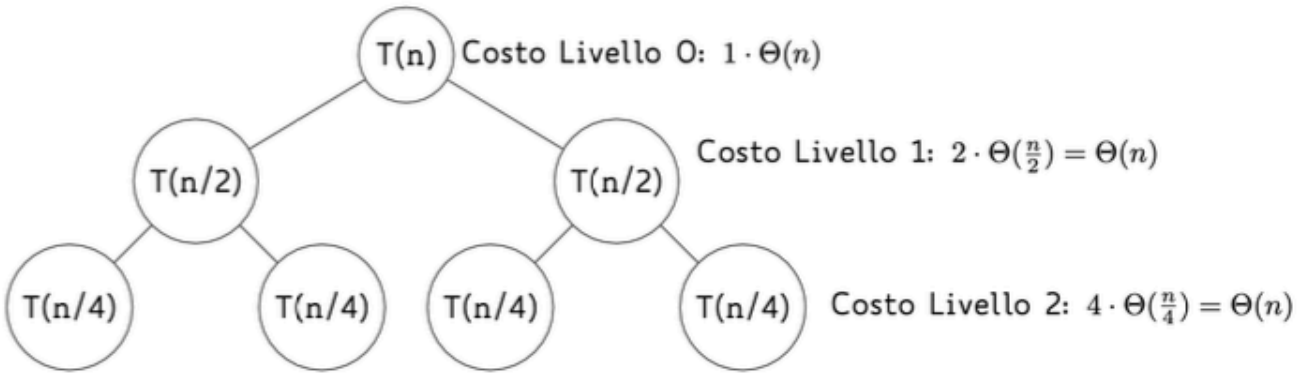
Costruzione dell'albero.

- Al livello 0, il costo è $\Theta(n)$.
- Al livello 1, abbiamo 2 sottoproblemi di dimensione $\frac{n}{2}$, ciascuno con costo $\Theta(\frac{n}{2})$, per un costo totale di $2 \cdot \Theta(\frac{n}{2}) = \Theta(n)$.
- Al livello 2, abbiamo 4 sottoproblemi di dimensione $\frac{n}{4}$, ciascuno con costo $\Theta(\frac{n}{4})$, per un costo totale di $4 \cdot \Theta(\frac{n}{4}) = \Theta(n)$.

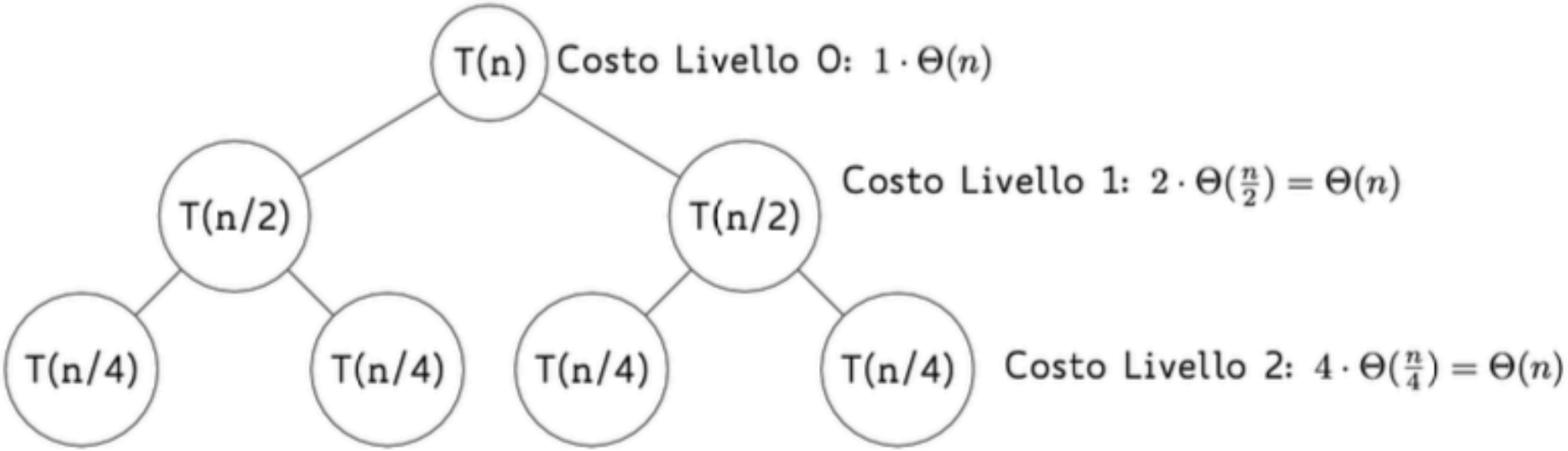
Al livello k, ci sono 2^k sottoproblemi di dimensione $\frac{n}{2^k}$, ciascuno con costo $\Theta\left(\frac{n}{2^k}\right)$, per un costo totale di:

$$\text{Costo livello } k = 2^k \cdot \Theta\left(\frac{n}{2^k}\right) = \Theta(n).$$

Nell'iimmagine che segue c'i sono i primi livelli dell' albero di ricorsione per $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$:



Seguito Esempio 2:



Calcolo del numero di livelli. Analogamente all'esempio precedente, il numero di livelli è $k = \log_2 n$.

Costo complessivo. Il costo totale è la somma dei costi su tutti i livelli:

$$\text{Costo totale} = \sum_{k=0}^{\log_2 n} \Theta(n) = \Theta(n \log n).$$

Esempio 3

Consideriamo la ricorrenza:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + \Theta(n) & \text{altrimenti} \end{cases}$$

Costruzione dell'albero. Analiziamo il costo di ciascun livello dell'albero:

- Al livello 0, il costo della radice è $\Theta(n)$.
- Al livello 1, abbiamo 2 sottoproblemi: il primo di dimensione $\frac{n}{2}$ con costo $\Theta(\frac{n}{2})$ ed uno di dimensione $\frac{n}{4}$ con costo $\Theta(\frac{n}{4})$. Il costo totale a livello 1 è $\left(\frac{1}{2} + \frac{1}{4}\right) \cdot \Theta(n) = \left(\frac{3}{4}\right) \Theta(n)$.
- Al livello 2, abbiamo 4 sottoproblemi.
  - Il nodo di dimensione $\frac{n}{2}$ genera due nodi di dimensioni $\frac{n}{4}$ e $\frac{n}{8}$, con costi $\Theta(\frac{n}{4})$ e $\Theta(\frac{n}{8})$ rispettivamente.
  - Il nodo di dimensione $\frac{n}{4}$ genera due nodi di dimensione $\frac{n}{8}$ e $\frac{n}{16}$ con costi $\Theta(\frac{n}{8})$ e $\Theta(\frac{n}{16})$ rispettivamente.

  Il costo totale a livello 2 è dunque

$$\left(\frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16}\right) \Theta(n) = \frac{9}{16}\Theta(n) = \left(\frac{3}{4}\right)^2 \Theta(n).$$

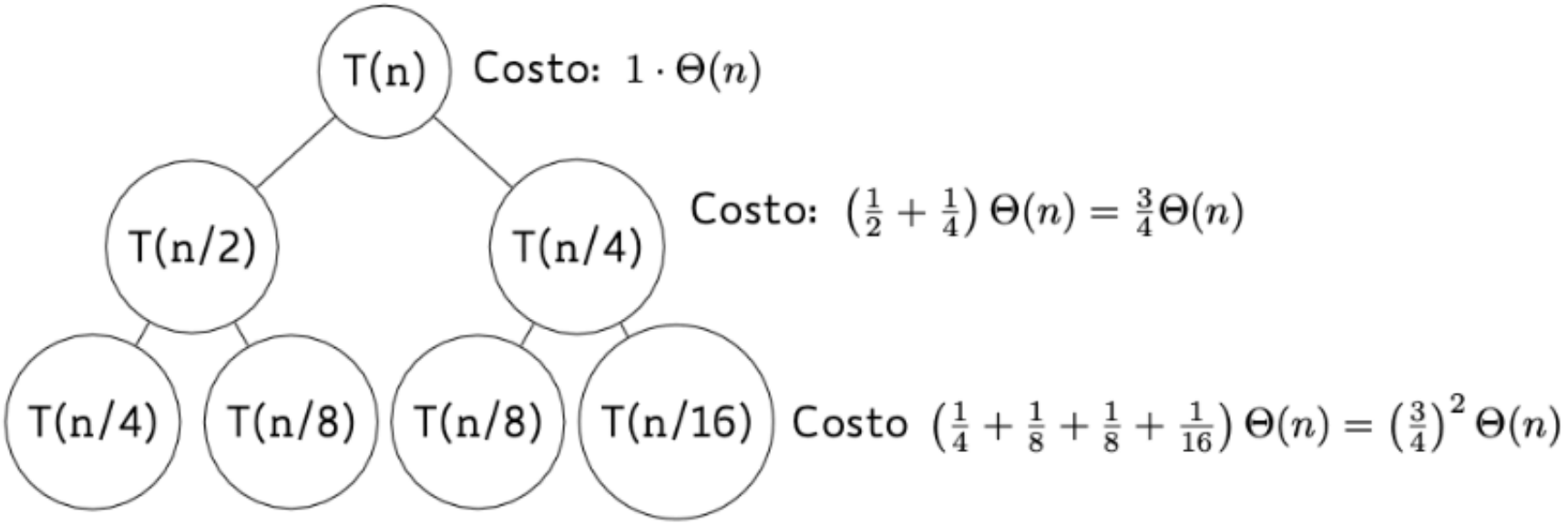
Seguito Esempio 3:

Costo al livello i ogni nodo di dimensione x al generico livello genera due figli, uno di dimensioni $\frac{x}{2}$ e uno di dimensione $\frac{x}{4}$. Il fattore complessivo di riduzione della dimensione e quindi anche dei costi è

$$\frac{1}{2}+\frac{1}{4}=\frac{3}{4}.$$

Di conseguenza, il costo totale al livello i è dato da $\left(\frac{3}{4}\right)^i \Theta(n)$.

Nell'immagine che segue ci sono i primi livelli dell' albero di ricorsione per $T(n)=T\left(\frac{n}{2}\right)+T\left(\frac{n}{4}\right)+\Theta(n)$:



Seguito Esempio 3:

Costo totale dell'albero Il costo totale dell'albero è dato dalla somma dei costi di tutti i livelli. Poiché il costo decresce geometricamente con il livello i, il costo totale è dominato dal primo livello (radice), che è $\Theta(n)$. Più formalmente: L'albero si espande fino a quando i sottoproblemi raggiungono dimensione 1, poiché la dimensione di ogni nodo si riduce di almeno $\frac{1}{4}$ per l'altezza h dell'albero si ha cioè:

$$\frac{n}{4^h} = 1 \implies h = \log_4 n.$$

Costo totale

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

La somma è una serie geometrica con ragione $\frac{3}{4} < 1$, quindi converge. Pertanto

$$T(n) \leq \Theta(1)\Theta(n) = \Theta(n)$$

Abbiamo dunque $T(n) = O(n)$. Inoltre banalmente $T(n) \geq \Theta(n)$.

Nota che allo stesso risultato si arriva con qualsiasi ricorrenza della forma:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ T(\alpha \cdot n) + T(\beta \cdot n) + \Theta(n) & \text{altrimenti} \end{cases}$$

dove $\alpha, \beta > 0$ e $\alpha + \beta < 1$.

Conclusione

Il metodo dell'albero consente di visualizzare e calcolare il costo totale di una ricorrenza distribuendo i costi attraverso i livelli della ricorsione. Questo approccio è particolarmente utile per ricorrenze semplici e per comprendere la struttura del problema ricorsivo.

Metodo di Sostituzione

Il metodo di sostituzione consiste nell'ipotesi di una soluzione esplicita per l'equazione di ricorrenza, seguita dalla verifica della validità di tale ipotesi mediante sostituzione nella ricorrenza stessa. Questo metodo è particolarmente utile per dimostrare che una data soluzione è corretta o per ottenere il comportamento asintotico della ricorrenza.

La procedura tipica del metodo di sostituzione comprende i seguenti passi:

1. Formulare un'ipotesi basata sull'osservazione della ricorrenza.
2. Sostituire l'ipotesi nella ricorrenza per verificare che sia soddisfatta.
3. Determinare eventuali costanti o parametri della soluzione.

Esempio 1

Consideriamo la seguente ricorrenza:

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

Supponiamo, per ipotesi, che $T(n) = \Theta(n)$.

Dimostreremo che $T(n) = \Theta(n)$ in due passi: prima mostrando che $T(n) \in O(n)$, e successivamente dimostrando che $T(n) \in \Theta(n)$.

Passo 1: Dimostrare che $T(n) \in O(n)$

Eliminando l'asintotica dall'equazione di ricorrenza, otteniamo:

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

Dimostreremo che esiste una costante c per cui $T(n) \leq cn$. Prendendo $c \geq b$ abbiamo

$$T(1) \leq b \leq c \cdot 1$$

e l'ipotesi vale per il caso base.
Applicando l'ipotesi di induzione, cioè $T(n-1) \leq c(n-1)$, otteniamo:

$$T(n) \leq c(n-1) + a = cn - (c-a) \leq cn$$

Dove l'ultima diseguaglianza vale prendendo $c \geq a$.
Quindi, prendendo $c = \max\{b, a\}$ abbiamo dimostrato che:

$$T(n) = O(n)$$

Passo 2: Dimostrare che $T(n) \in \Omega(n)$

Eliminando l'asintotica dall'equazione di ricorrenza, otteniamo:

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

Dimostreremo che esiste una costante c per cui $T(n) \geq cn$. Prendendo $c \leq b$ abbiamo

$$T(1) \geq b \geq c \cdot 1$$

e l'ipotesi vale per il caso base.
Supponiamo che l'ipotesi di induzione sia valida, cioè che $T(n-1) \geq c(n-1)$. Sostituendo questa ipotesi nella ricorrenza, otteniamo:

$$T(n) \geq c(n-1) + a = cn + (a - c) \geq cn$$

Dove l'ultima diseguaglianza vale prendendo $c \leq a$.
Quindi, prendendo $c = \min\{b, a\}$ abbiamo dimostrato che:

$$T(n) = \Omega(n)$$

Conclusione

Abbiamo dimostrato che $T(n) = O(n)$ e che $T(n) = \Omega(n)$, quindi possiamo concludere che:

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

Esempio 2

Consideriamo ora la ricorrenza:

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

Supponiamo, per ipotesi, che $T(n) = \Theta(n)$.

Dimostreremo che $T(n) = \Theta(n)$ in due passi: prima mostrando che $T(n) \in O(n)$, e successivamente dimostrando che $T(n) \in \Theta(n)$.

Passo 1: Dimostrare che $T(n) \in O(n)$

Eliminando l'asintotica dall'equazione di ricorrenza, otteniamo:

$$T(n) = \begin{cases} b & \text{se } n = 1 \\ 2T\left(\frac{n}{2}\right) + a & \text{altrimenti} \end{cases}$$

Dimostreremo che esistono due costanti costante c e d per cui $T(n) \leq cn - d$. Prendendo $c - d \geq b$ abbiamo

$$T(1) \leq b \leq c \cdot 1 - d$$

e l'ipotesi vale per il caso base.

Applicando l'ipotesi di induzione, cioè $T\left(\frac{n}{2}\right) \leq c\frac{n}{2} - d$ otteniamo:

$$T(n) \leq 2\left(c\frac{n}{2} - d\right) + a = cn - 2d + a = cn - d - (d - a) \leq cn - d$$

Dove l'ultima diseguaglianza vale prendendo $d \geq a$.

Quindi, prendendo $d \geq a$ e $c - d \geq b$ (come ad esempio $d = \max\{a, b\}$ e $c = 2d$) abbiamo dimostrato che:

$$T(n) = O(n)$$

Passo 2: Dimostrare che $T(n) \in \Omega(n)$

Eliminando l'asintotica dall'equazione di ricorrenza, otteniamo:

$$T(n) = \begin{cases} b & \text{se } n = 1 \\ 2T\left(\frac{n}{2}\right) + a & \text{altrimenti} \end{cases}$$

Dimostreremo che esiste una costante c per cui $T(n) \geq cn$. Prendendo $c \leq b$ abbiamo

$$T(1) = b \geq c \cdot 1$$

e l'ipotesi vale per il caso base.

Applicando l'ipotesi di induzione, cioè $T\left(\frac{n}{2}\right) \geq c\frac{n}{2}$ otteniamo:

$$T(n) \geq 2\left(c\frac{n}{2}\right) + a = cn + a \geq cn$$

Quindi, prendendo $c \leq b$ abbiamo dimostrato che:

$$T(n) = \Omega(n)$$

Conclusione

Abbiamo dimostrato che $T(n) = O(n)$ e che $T(n) = \Omega(n)$, quindi possiamo concludere che:

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

Teorema Principale

Il teorema principale delle ricorrenze fornisce un metodo per determinare la complessità asintotica di equazioni di ricorrenza del tipo:

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

dove:

- $a \geq 1$ è il numero di sottoproblemi, -
- $b > 1$ è il fattore di divisione del problema,
- $f(n)$ è un termine aggiuntivo che rappresenta il costo della divisione e della combinazione.

Il teorema si applica quando $f(n)$ è una funzione asintoticamente positiva e si basa sul confronto tra $f(n)$ e $n^{\log_b a}$, il termine dominante derivante dal lavoro ricorsivo. A seconda del confronto, si verificano tre casi:

1. **Caso 1:** $f(n) \in O\left(n^{\log_b a - \epsilon}\right)$ per qualche $\epsilon > 0$ In questo caso, il termine ricorsivo domina, e la soluzione è determinata da $n^{\log_b a}$:

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

Esempio:

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

Calcoliamo $\log_b a = \log_2 2 = 1$. Poiché $f(n) = \Theta(1) \in O(n^{1-\epsilon})$ per $\epsilon = 1$, il caso 1 si applica. La soluzione è:

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

2. **Caso 2:** $f(n) \in \Theta\left(n^{\log_b a}\right)$ In questo caso, il termine ricorsivo e il termine $f(n)$ sono bilanciati, e la soluzione è determinata da $n^{\log_b a} \log n$:

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

Esempio:

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

Calcoliamo $\log_b a = \log_2 2 = 1$. Poiché $f(n) = \Theta(n) = \Theta(n^{\log_b a})$, il caso 2 si applica. La soluzione è:

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

3. **Caso 3:** $f(n) \in \Omega\left(n^{\log_b a+\epsilon}\right)$ per qualche $\epsilon > 0$, e la **condizione di regolarità** $af(n/b) \leq cf(n)$ per una costante $c < 1$ è soddisfatta In questo caso, il termine $f(n)$ domina, e la soluzione è determinata interamente da $f(n)$:

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

Esempio:

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

Calcoliamo $\log_b a = \log_2 1 = 0$. Poiché $f(n) = \Theta(n^2) \in \Omega(n^{0+\epsilon})$ con $\epsilon = 2$, e la condizione di regolarità è soddisfatta, il caso 3 si applica. La soluzione è:

$$T(n) = \Theta(n^2).$$

Conclusione Il teorema principale offre un metodo rapido per risolvere molte equazioni di ricorrenza comuni, classificandole in tre casi basati sul confronto tra $f(n)$ e $n^{\log_b a}$. Tuttavia, è importante verificare che la ricorrenza soddisfi i requisiti del teorema prima di applicarlo.

I tre casi del Teorema Principale non sono esaustivi:

- ci sono funzioni $f(n)$ che sono dominate ma non fortemente dominate.
- ci sono funzioni $f(n)$ che sono dominanti ma non fortemente dominanti.

In questi casi, il Teorema Principale non è applicabile e bisogna utilizzare metodi alternativi come l'iterazione o la sostituzione per risolvere la ricorrenza.

Esempio 1: il Caso 1 non è applicabile

Consideriamo la ricorrenza:

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

Analisi. Confrontiamo $f(n)$ con $n^{\log_b a}$, dove $a = 2$ e $b = 2$, quindi $n^{\log_b a} = n$. Abbiamo che:

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

Osserviamo che:

$$f(n) \in O(n) \quad \text{ma} \quad f(n) \notin O\left(\frac{n}{n^\epsilon}\right) \quad \text{per ogni } \epsilon > 0.$$

Pertanto, $f(n)$ è dominato da n, ma non è fortemente dominato.

Esempio 2: il Caso 3 non è applicabile

Consideriamo la ricorrenza:

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

Analisi. Confrontiamo $f(n)$ con $n^{\log_b a}$, dove $a = 2$ e $b = 2$, quindi $n^{\log_b a} = n$. Abbiamo che:

$$f(n) = n \log n, \quad n^{\log_b a} = n.$$

Osserviamo che:

$$f(n) \in \Omega(n) \quad \text{ma} \quad f(n) \notin \Omega(n^{1+\epsilon})$$

Pertanto, $f(n)$ domina il termine ricorsivo, ma non lo domina fortemente.

Approssimazione di una Ricorrenza

In alcuni casi, le ricorrenze complesse possono essere approssimate superiormente e inferiormente per ottenere stime sull'andamento asintotico della soluzione. Consideriamo la seguente ricorrenza:

$$T(n = \begin{cases} \Theta(1) & \text{se } n \leq 2 \\ T(n-1) + T(n-2) + \Theta(1) & \text{altrimenti} \end{cases}$$

Limitazione Inferiore

Possiamo approssimare la ricorrenza inferiormente sostituendo $T(n-1)$ con il termine inferiore $T(n-2)$:

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

Questa nuova ricorrenza rappresenta un limite inferiore per $T(n)$. Espandiamo la ricorrenza iterativamente:

$$\begin{aligned} T(n) &\geq 2T(n-2) + \Theta(1) \\ &\geq 2\left(2T(n-4) + \Theta(1)\right) + \Theta(1) \\ &= 2^2 T(n-4) + 2\Theta(1) + \Theta(1) \\ &= \ldots\ldots \\ &\geq 2^k T(n-2k) + \sum_{i=0}^{k-1} 2^i \cdot \Theta(1) \end{aligned}$$

Per $k = \lfloor n/2 \rfloor$, otteniamo $T(n-2k) = T(1) = \Theta(1)$. Quindi:

$$T(n) \geq 2^{\lfloor n/2 \rfloor} \cdot \Theta(1) + \sum_{i=0}^{\lfloor n/2 \rfloor - 1} 2^i \cdot \Theta(1).$$

La somma geometrica si risolve come:

$$\sum_{i=0}^{\lfloor n/2 \rfloor - 1} 2^i = 2^{\lfloor n/2 \rfloor} - 1.$$

Pertanto:

$$T(n) \geq 2^{\lfloor n/2 \rfloor} \cdot \Theta(1) = \Theta(2^{n/2}).$$

Limitazione Superiore

Limitiamo la ricorrenza superiormente sostituendo $T(n-2)$ con il termine maggiore $T(n-1)$:

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

Espandendo iterativamente:

$$\begin{aligned} T(n) &\leq 2T(n-1) + \Theta(1)) \\ &\leq 2\left(2T(n-2) + \Theta(1)\right) + \Theta(1) \\ &= 2^2 T(n-2) + 2\Theta(1) + \Theta(1) \\ &= \ldots\ldots \\ &\leq 2^k T(n-k) + \sum_{i=0}^{k-1} 2^i \cdot \Theta(1) \end{aligned}$$

Per $k = n-1$, otteniamo $T(n-k) = T(1) = \Theta(1)$. Quindi:

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

La somma geometrica si risolve come:

$$\sum_{i=0}^{n-2} 2^i = 2^{n-1} - 1.$$

Pertanto:

$$T(n) \leq 2^{n-1} \cdot \Theta(1) = \Theta(2^n).$$

Conclusione

Abbiamo stabilito i seguenti limiti per $T(n)$:

$$\Theta(2^{n/2}) \leq T(n) \leq \Theta(2^n).$$

La funzione $T(n)$ cresce dunque esponenzialmente, con un comportamento compreso tra $2^{0.5n}$ e 2^n.
Con un analisi più precisa si può ottenere $T(n) \approx 2^{0.694n}$

Corso di laurea in Informatica
Introduzione agli Algoritmi

Esercizi per casa

SAPIENZA
UNIVERSITÀ DI ROMA

Calcolare la soluzione delle seguenti equazioni di ricorrenza:

1. $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta\left(n^{\log_2 3}\right)$
2. $T(n) = T(n-1) + \Theta(2^n)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta\left(2^n\right)$
3. $T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n^2)$
4. $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n^3)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n^3)$
5. $T(n) = 16T\left(\frac{n}{2}\right) + \Theta(n^2)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n^4)$
6. $T(n) = 2T(n-1) + \Theta(n)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(2^n)$
7. $T(n) = \sqrt{n}T(\sqrt{n}) + \Theta(n)$ $T(2) = \Theta(1)$ soluzione: $T(n) = \Theta(n \log\log n)$
8. $T(n) = 9T(n-3) + \Theta(2^n)$ $T(0) = \Theta(1)$ soluzione: $T(n) = \Theta(9^{\frac{n}{3}})$
9. $T(n) = T(n-1) + \Theta(\sqrt{n})$, $T(0) = \Theta(1)$ soluzione: $T(n) = \Theta(n^{\frac{3}{2}})$
10. $T(n) = T(\sqrt{n}) + \Theta(1)$, $T(2) = \Theta(1)$ soluzione: $T(n) = \Theta(\log\log n)$
11. $T(n) = T\left(\frac{n}{2}\right) + \Theta(n^2)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n^2)$
12. $T(n) = 2T\left(\frac{n}{2}\right) + \Theta\left(\frac{n}{\log n}\right)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n \log\log n)$
13. $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n \log n)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n \log^2 n)$
14. $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + \Theta(1)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n)$
15. $T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + \Theta(n)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n \log n)$
16. $T(n) = n \cdot T(n-1) + \Theta(n!)$, $T(1) = \Theta(1)$ soluzione: $T(n) = \Theta(n \cdot n!)$