Esercizi: analisi asintotica delle funzioni di complessità ricorsive

January 31, 2007

Il Metodo di Sostituzione: esercizi risolti

Si utilizzi il metodo di sostituzione per studiare le seguenti ricorrenze (per le ricorrenze $2, 3$ viene suggerito a destra l'ordine di grandezza).

$$T(n) = \begin{cases} T(n-1) + \Theta(n) & \text{se } n > 1, \\ \Theta(1) & \text{se n=1.} \end{cases} \tag{1}$$

$$T(n) = \begin{cases} T(\lfloor \frac{2n}{3} \rfloor) + \Theta(1) & \text{se } n > 1, \\ \Theta(1) & \text{se n=1.} \end{cases} \qquad [T(n) = \Theta(\log n)] \tag{2}$$

$$T(n) = \begin{cases} 4T(\frac{n}{2}) + \mathcal{O}(n^2) & \text{se } n > 1, \\ \Theta(1) & \text{se } n \leq 1. \end{cases} \qquad [T(n) = \mathcal{O}(n^2 \log n)] \tag{3}$$

$$T(n) = \begin{cases} T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + 1 & \text{se } n > 1, \\ \Theta(1) & \text{se n=1.} \end{cases} \tag{4}$$

Il metodo di sostituzione prevede due passi. Nel primo passo si stima l'ordine di grandezza asintotico per $T(n)$. Nel secondo passo si dimostra, per induzione su n, la correttezza dell'ordine di grandezza stimato. Il metodo si rivela utile quando si ha già un'idea della soluzione alla ricorrenza studiata[a].

[a] Per stimare l'ordine di grandezza di una funzione di complessità ricorsiva è necessario avere un minimo di "background" ed esperienza: dunque bisogna aver fatto un po' di esercizi e bisogna destreggiarsi bene con le ricorrenze che esprimono la complessità di algoritmi noti

Soluzione.

(1) **Guess:** La funzione di complessità data è la funzione di complessità della versione ricorsiva di selectionsort. Dunque stimeremo $T(n) = \Theta(n^2)$.

Induzione. Provo per induzione su n che $\forall n \geq n_0 > 0$:

1. $T(n) \geq c_1 n^2$ (cioe' $T(n) = \Omega(n^2)$)

2. $T(n) \leq c_2 n^2$ (cioe' $T(n) = \mathcal{O}(n^2)$)

dove c_1 e c_2 sono costanti positive.

1. – *Base:* per $n_0 = 1$, $T(1) = \Theta(1) \geq c_1(1^2)$ a patto di scegliere una frazione positiva c_1 abbastanza piccola
   – *Passo Induttivo*
   * *Ipotesi Induttiva:* $T(n') \geq c_1(n')^2$ per ogni $1 \leq n' < n$
   * Provo che $T(n) \geq c_1 n^2$.

$$\begin{array}{lll} T(n) & = T(n-1) + \Theta(n) & \text{usando la relazione ricorsiva} \\ & \geq T(n-1) + b_1 n & \Theta(n) \text{ rappresenta } b_1 n \leq f(n) \leq b_2 n \\ & \geq c_1(n-1)^2 + b_1 n & \text{ipotesi induttiva} \\ & \geq c_1 n^2 + (b_1 - 2c_1)n + c_1 & \\ & \geq c_1 n^2 + (b_1 - 2c_1)n & \\ & \geq c_1 n^2 & \text{per } 0 < c_1 \leq \frac{b_1}{2} \end{array}$$

2. – *Base:* Per $n_0 = 1$, $T(1) = \Theta(1) \leq c_2(1^2)$ a patto di scegliere c_2 abbastanza grande.
   – *Passo Induttivo*
   * *Ipotesi Induttiva:* $T(n') \leq c_2(n')^2$ per ogni $1 \leq n' < n$
   * Provo che $T(n) \leq c_2 n^2$.

$$\begin{array}{lll} T(n) & = T(n-1) + \Theta(n) & \text{usando la relazione ricorsiva} \\ & \leq T(n-1) + b_2 n & \Theta(n) \text{ rappresenta } b_1 n \leq f(n) \leq b_2 n \\ & \leq c_2(n-1)^2 + b_2 n & \text{ipotesi induttiva} \\ & \leq c_2 n^2 + (b_2 - 2c_2)n + c_2 & \\ & \leq c_2 n^2 & \text{per } c_2 \geq b_2 \end{array}$$

L'ultima disuguaglianza segue dal fatto che, su $n \geq n_0 = 1$, la retta $(b_2 - 2c_2)n + c_2$ assume valori ≤ 0 per $c_2 \geq b_2$. Infatti, da una parte il segno della derivata ci assicura che la retta è decrescente per $c_2 \geq \frac{b_2}{2}$. Dall'altra, per $n_0 = 1$ si ha $(b_2 - 2c_2)1 + c_2 \leq 0$ se e solo se $c_2 \geq b_2$.

(2) **Guess:** il testo dell'esercizio suggerisce che $T(n) = T(\lfloor \frac{2n}{3} \rfloor) + \Theta(1)$ stia in $\Theta(lg(n))$.

Induzione. Provo per induzione su n che $\forall n \geq n_0 > 0$:

1. $T(n) \leq c_1 \cdot lg(n)$ (cioe' $T(n) = \mathcal{O}(lg(n))$)
2. $T(n) \geq c_2 \cdot lg(n)$ (cioe' $T(n) = \Omega(lg(n))$)

dove c_1 e c_2 sono costanti positive.

1. – *Base:* per $n_0 = 2$, $T(2) = \Theta(1) \leq c_1 \cdot 2lg(2)$ a patto di sciegliere c_1 abbastanza grande [1]
   – *Passo Induttivo*
   * *Ipotesi Induttiva:* $T(n') \leq c_1 \cdot lg(n')$ per ogni $2 \leq n' < n$
   * Provo che $T(n) \leq c_1 \cdot lg(n)$.

$$
\begin{array}{lll}
T(n) & = T(\lfloor \frac{2n}{3} \rfloor) + \Theta(1) & \text{usando la relazione ricorsiva} \\
& \leq T(\lfloor \frac{2n}{3} \rfloor) + b & \Theta(1) \text{ nasconde una costante} \\
& \leq c_1 lg(\lfloor \frac{2n}{3} \rfloor) + b & \text{ipotesi induttiva} \\
& \leq c_1 lg(\frac{2n}{3}) + b & \\
& \leq c_1 lg(n) + c_1 lg(\frac{2}{3}) + b & \text{propieta' dei logaritmi} \\
& \leq c_1 lg(n) & \text{basta aver scelto } c_1 \text{ abbastanza grande da far prevalere il termine negativo } c_1 lg(\frac{2}{3}) \text{ sul ter. pos. } b
\end{array}
$$

2. – *Base:* per $n_0 = 2$, $T(2) = \Theta(1) \geq c_2 \cdot 2lg(2)$ a patto di sciegliere una frazione positiva c_2 abbastanza piccola
   – *Passo Induttivo*
   * *Ipotesi Induttiva:* $T(n') \geq c_2 \cdot lg(n')$ per ogni $2 \leq n' < n$
   * Provo che $T(n) \geq c_2 \cdot lg(n)$.

[1] Non potevo sciegliere $n_0 = 1$ perchè $lg1 = 0$ e dunque $T(1) \not\leq c_1 \cdot lg1$ per nessuna $c_1 > 0$. Si noti che avendo scielto $n_0 = 2$ non riesco a trattare nel passo induttivo $T(3) = T(1) + \Theta(1)$ perchè dipende da $T(1 < n_0 = 2)$. Tuttavia basta aver scelto c_1 grande a sufficienza per far valere $T(3) \leq c \cdot lg(3)$. In generale basta sempre scegliere c_1 grande a sufficienza per far valere la tesi sui $T(n)$ che dipendono da $T(n' < n_0)$.

$$
\begin{array}{lll}
T(n) & = T(\lfloor \frac{2n}{3} \rfloor) + \Theta(1) & \text{usando la relazione ricorsiva} \\
& \geq c_2 lg(\lfloor \frac{2n}{3} \rfloor) + b & \text{ipotesi induttiva} \\
& \geq c_2 lg(\frac{2n}{3} - 1) + b & \\
& = c_2 lg(\frac{2n-3}{3}) + b & \\
& \geq c_2 lg(\frac{n}{2} \cdot \frac{1}{3}) + b & 2n - 3 \geq \frac{n}{2} \text{ per } n \geq 2 \qquad (*) \\
& = c_2 lg(n) - c_2 lg(6) + b & \text{propieta' logaritmi} \\
& \geq c_2 lg(n) & \text{se frazione positiva } c_2 \text{ è abbastanza piccola da far prevalere il termine positivo } b \text{ sul ter. neg. } -c_2 lg(6)
\end{array}
$$

NOTA Specificando che le funzioni di *floor* ($\lfloor\ \rfloor$) e *ceiling* ($\lceil\ \rceil$) non hanno effetto asintoticamente ci si poteva ridurre a studiare $T(n) = T(\frac{2}{3}n) + \Theta(1)$ evitando il passaggio in $(*)$.

(3) **Guess** Il testo dell'esercizio suggerisce $T(n) = \mathcal{O}(n^2)lg(n)$. Si noti che, poichè in $T(n) = 4T(\frac{n}{2}) + \mathcal{O}(n^2)$ compare un $\mathcal{O}()$ è possibile dare solo una stima dall'alto ("con $\mathcal{O}$").

Induzione Provo per induzione su n che $\forall n \geq n_0 > 0$:
$T(n) \leq cn^2 lg(n)$ (cioe' $T(n) = \mathcal{O}(n^2 lg(n))$)
dove c è una costante positiva opportuna.

- *Base* Per $n_0 = 2$, $T(2) = \Theta(1) \leq c \cdot 4lg(2)$ a patto di sciegliere c abbastanza grande.
- *Passo Induttivo*
  - *Ipotesi Induttiva* $T(n') \leq c(n')^2 lg(n')$ per ogni $1 \leq n' < n$
  - Provo che $T(n) \leq c(n)^2 lg(n)$.

$$
\begin{array}{lll}
T(n) & = 4T(\frac{n}{2}) + \mathcal{O}(n^2) & \text{usando la relazione ricorsiva} \\
& \leq 4T(\frac{n}{2}) + bn^2 & \mathcal{O}(n^2) \text{ rappresenta } f(n) \leq cn^2 \\
& \leq 4c\frac{n^2}{4} lg(\frac{n}{2}) + bn^2 & \text{ipotesi induttiva} \\
& \leq cn^2 lg(n) - cn^2 lg(2) + bn^2 & \text{propieta' dei logaritmi} \\
& \leq cn^2 lg(n) & \text{per } c \geq \frac{b}{lg(2)}
\end{array}
$$

(4) **Guess** La funzione di complessita' data descrive un algoritmo che, per risolvere un problema di dimensione n, si riduce a risolvere due sottoproblemi di dimensione $\frac{n}{2}$, come MergeSort. Pero', mentre MergeSort spende $\Theta(n)$ per comporre le soluzioni ai sottoproblemi, nel nostro caso il costo esterno alla ricorsione è 1. Dunque stimiamo una complessita' un po' piu' bassa di quella di MergeSort: $T(n) = \Theta(n)$.

Induzione Provo per induzione su n che $\forall n \geq n_0 > 0$:

1. $T(n) \leq c_2 n$ (cioe' $T(n) = \mathcal{O}(n)$)
2. $T(n) \geq c_1 n$ (cioe' $T(n) = \Omega(n)$)

dove c_1 e c_2 sono costanti positive.

1. – *Base* Per $n_0 = 1$, $T(1) = \Theta(1) \leq c_2$ per c_2 opportuna.
   – *Passo Induttivo*
   * *Ipotesi Induttiva* $T(n') \leq c_2 n'$ per ogni $1 \leq n' < n$
   * Provo che $T(n) \leq c_2 n$.

$$\begin{array}{lll} T(n) & = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1 & \text{usando la relazione ricorsiva} \\ & \leq c_2 n + 1 & \text{ipotesi induttiva} \\ & \not\leq c_2 n & \text{??? Non Funziona !!!} \quad (*) \end{array}$$

Quando il passo induttivo non funziona ci sono due possibilita'. La prima è che la mia stima dell'ordine di grandezza di $T(n)$ non sia corretta (e non è questo il caso). Oppure, ho stimato bene $T(n)$ *ma la mia ipotesi induttiva è troppo debole*. In questo caso rafforzando l'ipotesi induttiva e assumendo la condizione $T(n') \leq cn - b$ (piu' forte di $T(n') \leq cn$) riusciamo a portare a termine il passo induttivo ottenendo

$$\forall n \geq 1 (T(n) \leq cn - b) \Rightarrow \forall n \geq 1 (T(n) \leq cn))$$

Un indizio del fatto che bisogna rafforzare l'ipotesi induttiva (e non cambiare l'ordine di grandezza stimato) e' dato dal fatto che in $(*)$ ci avanzano termini di ordine di grandeza inferiore a quello stimato. Non sempre è facile

rafforzare opportunamente l'ipotesi induttiva ...

$$\begin{array}{ll} T(n) & = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1 \quad \text{usando la relazione ricorsiva} \\ & \leq c_2 n - 2b + 1 \quad \text{ipotesi induttiva} \\ & \leq c_2 n - b \end{array}$$

2. – *Base* Per $n_0 = 1$, $T(1) = \Theta(1) \geq c_1(1)$ a patto di sciegliere una frazione positiva c_1 abbastanza piccola
   – *Passo Induttivo*
     * *Ipotesi Induttiva* $T(n') \geq c_1(n')$ per ogni $1 \leq n' < n$
     * Provo che $T(n) \geq c_1 n$.

$$\begin{array}{ll} T(n) & = T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1 \quad \text{usando la relazione ricorsiva} \\ & \geq c_1 n + 1 \quad \text{ipotesi induttiva} \\ & \geq c_1 n \end{array}$$

Quando non si hanno idee precise sull'ordine di grandezza di una funzione di complessita' ricorsiva, sono utili i metodi di Svolgimento della Ricorsione (o Metodo dell'Iterazione) e di Albero Di Ricorsione. Entrambi i metodi permettono di "capire come si dipanano i costi nella ricorsione". Nel *Metodo di Iterazione* si *costruisce una sommatoria di costi* svolgendo la ricorsione e ci si riduce allo studio di tale sommatoria. L'*Albero di Ricorsione* permette invece di visulaizzare in un albero il dipanarsi dei costi delle chiamate ricorsive.

Il Metodo di Iterazione: esercizi risolti

Si studino le seguenti ricorrenze con il metodo di *Svolgimento della Ricorsione*

1. $T(n) = T(\lfloor \frac{n}{c} \rfloor) + \Theta(1)$ dove $c > 1$
2. $T(n) = T(n-1) + \Theta(lg(n))$

In entrambe le ricorrenze si assuma $T(1) = \Theta(1)$.

Soluzione Il metodo dello *Sviluppo dell'Iterazione* prevede che si svolga la ricorsione in una sommatoria e si studi quindi la sommatoria ottenuta.

1. Sviluppo l'iterazione e studio la sommatoria.

$T(n) = T(\lfloor \frac{n}{c} \rfloor) + \Theta(1) =$
$= (T(\lfloor \frac{1}{c} \lfloor \frac{1}{c} n \rfloor \rfloor) + \Theta(1)) + \Theta(1) =$
$= ((T(\lfloor (\frac{1}{c})^2 n \rfloor) + \Theta(1)) + \Theta(1)) =$
$= ((T(\lfloor (\frac{1}{c})^3 n \rfloor) + \Theta(1)) + \Theta(1)) + \Theta(1) =$
$= \cdots =$
$= T(1) + \log_c(n)(\Theta(1)) = \Theta(lg(n))$

Ad ogni passo dello sviluppo moltiplico per $\frac{1}{c}$ l'argomento della funzione di complessita' T. Mi fermo quando $(\frac{1}{c})^k n = 1$ cioe' dopo $k = \log_c(n)$ passi. Si ricordi inoltre che $\log_c(n) = \Theta(\log(n))$ poichè la base del logaritmo non conta asintoticamente (si veda il problema 2 della prima dispensa di esercizi). Infine si noti che, ad esempio, sia la funzione di complessita' del caso peggiore di *Heapify* ($T(n) \leq T(\lfloor \frac{2}{3} n \rfloor + \Theta(1))$, sia la complessita' della *Ricerca Binaria* ($T(n) \leq T(\lfloor \frac{1}{2} n \rfloor + \Theta(1))$ sono un caso particolare della funzione di complessita' appena studiata. Sia *Heapify* che *Ricerca Binaria* hanno complessita' $\mathcal{O}(\log(n))$. Un'ultima osservazione: potevo assumere $n = c^k$ per liberarmi del $\lfloor \ \rfloor$.

2. Sviluppo l'iterazione e studio la sommatoria. Esplicito la costante "nascosta" in $\Theta(\log(n)$ scrivendo $c\log(n)$
$T(n) = T(n-1) + c\log(n) =$
$= (T(n-2) + c\log(n-1)) + c\log(n) =$
$= \cdots =$
$= T(1) + c\log(2) + \cdots + c\log(n) =$ (♣)
$= \Theta(1) + c(\sum_{i=2}^{n} lg(i)) =$
$= \Theta(n \cdot \log(n))$.

Infatti, $\sum_{i=2}^{n} lg(i) = \Theta(n(lg(n))$. Dimostrare che $\sum_{i=2}^{n} \log i = \mathcal{O}(n\log(n))$ è banale (basta maggiorare ogni termine della sommatoria con il termine $\log n$). Un suggerimento per provare che $\sum_{i=2}^{n} lg(i) = \Omega(n \log n)$ e' quello di notare che ci sono almeno $\frac{n}{2}$ termini nella sommatoria che superano $lg(\frac{n}{2})$. Se i suggerimenti non sono sufficienti a portare a termine la dimostrazione si consulti l'esercizio 2 della prima dispensa.

Si noti l'abuso di notazione se in♣ non esplicitavo la costante nascosta in Θ scrivendo $\sum_{i=1}^{\log(n)} \Theta(\log(i))$. In tal caso i termini della sommatoria si interpretano come il valore assunto in i dalla *medesima* funzione $c_1 lg(n) \leq f(n) \leq c_2 lg(n)$ e vale $\sum_{i=1}^{\log(n)} \Theta(\log(i)) = \Theta(\sum_{i=1}^{\log(n)} \log(i))$.

Il metodo dell'Albero di Ricorsione: esercizi risolti

3-a Si analizzino mediante l'*Albero di ricorsione* le seguenti ricorrenze:

1. $T(n) = 3T(\frac{n}{4}) + n$
2. $T(n) = 3T(\frac{n}{3}) + n$
3. $T(n) = 3T(\frac{n}{2}) + n$

3-b Si analizzino mediante l'*Albero di ricorsione* le seguenti ricorrenze:

1. $T(n) = T(\frac{n}{5}) + T(\frac{3}{4}n) + \Theta(n)$
2. $T(n) = T(\frac{n}{4}) + T(\frac{3}{4}n) + \Theta(n)$

   Si noti che la prima delle due funzioni di complessità è "approssimativamente" l'equazione di complessità per Order Statistics e l'albero di ricorsione mostra che sta in $\Theta(n)$. La seconda delle equazioni sarebbe l'equazione che si otterrebbe se in Order Statistics si trovasse la mediana di un gruppo di $\frac{n}{4}$ elementi anzichè di un gruppo di $\frac{n}{5}$ elementi. L'albero di ricorsione mostra che quest'ultima ricorrenza è $\Theta(n \log n)$.

3-c Si analizzi con l'albero di ricorsione $T(n) = 3T(\frac{n}{4}) + n^2$

3-d Si analizzi con l'albero di ricorsione $T(n) = aT(\frac{n}{b}) + n^p$ dove a è un intero maggiore o uguale ad 1, $b > 1, p > 0$ (b, p possono essere anche frazioni).

Soluzione L'Albero di Ricorsione permette di visualizzare lo sviluppo della ricorsione e dei costi mediante un albero. Ogni nodo interno dell'albero ha lo stesso numero di figli e quest'ultimo dipende dal numero di chiamate ricorsive (ex. nelle ricorrenze in 3-b ho 2 chiamate ricorsive. Nelle ricorrenze in 3-a ho 3 chiamate ricorsive).
Ogni nodo nell'albero di ricorsione per $T(n) = p \cdot T(m) + f(n)$ rappresenta una chiamata ricorsiva su un input di dimensioni $l \leq n$ ed è associato al costo $f(l)$: $f(l)$ è il costo di $T(l)$ a meno delle chiamate ricorsive.
Per valutare $T(n)$ si calcolano i costi associati ad ogni *livello* dell'albero di ricorsione per $T(n)$, si calcola l'*altezza* dell'albero, ed infine si perviene ad una stima globale di $T(n)$.

3-a 1. L'albero di Ricorsione per $T(n) = 3T(\frac{n}{4}) + n$ è rappresentato in Figura 1. Dato il nodo rappresentante $T(m)$, vi ho associato solo il costo $f(m) = m$ senza specificare la dimensione dell'input m (in questo caso dimensione dell'input e funzione di costo esterno alle chiamate ricorsive coincidono). A destra di ogni livello ho calcolato il costo totale del livello stesso. Si noti che l'albero è completo (ogni ramo ha la stessa lunghezza perche in ogni ramo divido ogni volta l'input per 4). L'altezza dell'albero è $\log_4(n)$ ed il numero di foglie è $3^{\log_4 n} = n^{\log_4 3}$. Dato l'Albero di Ricorsione in figura 1 mostriamo che $T(n) = \mathcal{O}(n)$.

$T(n) \leq n \cdot \sum_{i=0}^{log_4(n)} (\frac{3}{4})^i$

$\leq n \cdot \sum_{i=0}^{\infty} (\frac{3}{4})^i = n\frac{1}{(1-\frac{3}{4})} \Rightarrow T(n) = \mathcal{O}(n)$

Per ottenere il risultato sopra si è sfruttato il fatto che $\sum_{i=0}^{\infty} (\frac{3}{4})^i$ converge a $\frac{1}{(1-\frac{3}{4})}$ poichè e' una sommatoria geometrica di ragione $\frac{3}{4} < 1$.

Banalmente, $T(n) = \Omega(n)$ poichè $T(n) \geq n$. Se ne conclude che $T(n) = \Theta(n)$.

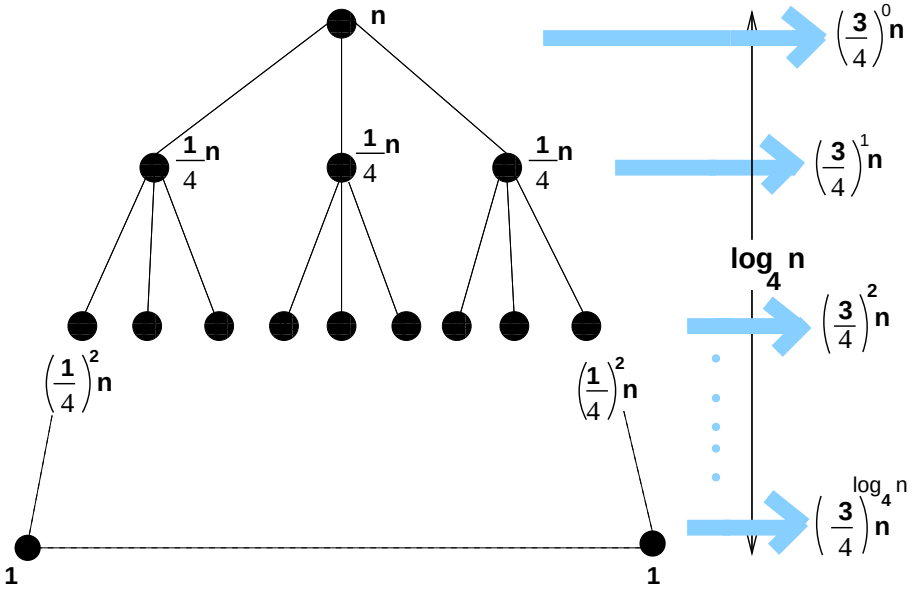

Figure 1: Albero di Ricorsione per $T(n) = 3T(\frac{n}{4}) + n$

2. La figura 2 rappresenta l'albero di ricorsione per $T(n) = 3T(\frac{n}{3})+n$.

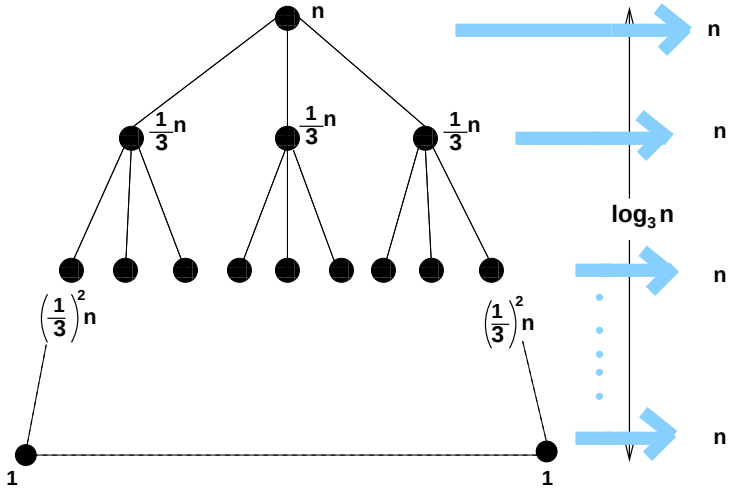

Figure 2: Albero di Ricorsione per $T(n) = 3T(\frac{n}{3}) + n$

L'albero ha altezza $log_3 n$ ed ha $3^{log_3 n} = n^{log_3 3} = n$ foglie. Dato l'albero di ricorsione per $T(n) = 3T(\frac{n}{3}) + n$ è possibile provare che $T(n) = \Theta(n \log n)$.

$$T(n) = (\log_3 n + 1) \cdot n = \Theta(n \log n)$$

3. La figura 3 rappresenta l'albero di ricorsione per $T(n) = 3T(\frac{n}{2})+n$.

L'albero ha altezza $\log_2 n$ ed ha $n^{\log_2 3}$ foglie. Dato l'albero di ricorsione per $T(n) = 3T(\frac{n}{2}) + n$ è possibile provare che $T(n) = \Theta(n^{\log_2 3})$

$T(n) = n \cdot \sum_{i=0}^{\log_2 n} (\frac{3}{2})^i = n \cdot \frac{\frac{3}{2}^{\log_2 n+1}-1}{\frac{3}{2}-1} = n \cdot \frac{\frac{3}{2} \cdot \frac{3^{\log_2 n}}{2^{\log_2 n}}-1}{\frac{3}{2}-1} =$
$= n \cdot \frac{\frac{3}{2} \cdot \frac{n^{\log_2 3}}{n}-1}{\frac{3}{2}-1} =$
$= \frac{\frac{3}{2} \cdot n^{\log_2 3}-n}{\frac{3}{2}-1} =$
$= \Theta(n^{\log_2 3})$

3-b 1. L'albero di Riorsione che si ottiene per $T(n) = T(\frac{n}{5}) + T(\frac{3}{4}n) + \Theta(n)$ è rappresentato in Figura 4. Dato il nodo rappresentante

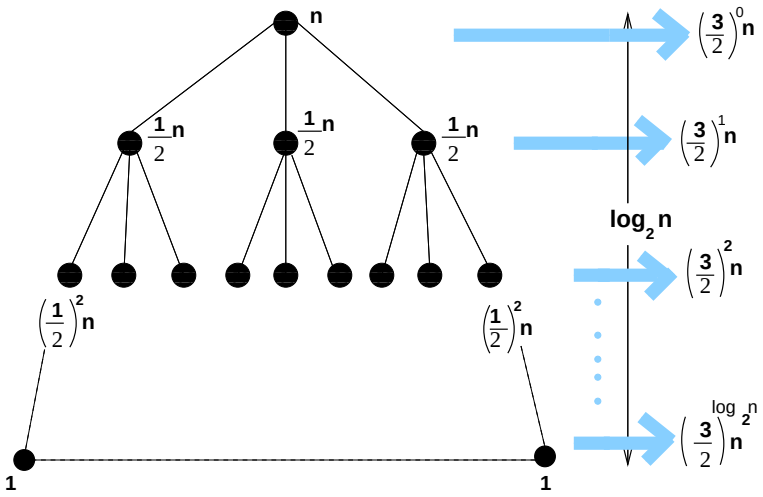

Figure 3: Albero di Ricorsione per $T(n) = 3T(\frac{n}{2}) + n$

$T(m)$, vi ho associato il costo $f(m) = cm$ esplicitando con $c \cdot m$ la costante sottointesa da $\Theta(m)$. Accanto ad ogni nodo, in corsivo, ho specificato la dimensione dell'input (su cui avviene la chiamata ricorsiva rappresentata dal nodo). A destra di ogni livello ho calcolato il costo totale del livello stesso. Nel ramo di sinistra la dimensione dell'input viene divisa ogni volta per 5: la lunghezza di tale ramo è dunque $log_5 n$. Nel ramo di destra la dimensione dell'input viene divisa ogni volta per $\frac{4}{3}$: la lunghezza di tale ramo è dunque $log_{\frac{4}{3}} n$. Nei rami in mezzo divido un po' per 5 e un po' per $\frac{4}{3}$, scendendo a sinistra o a destra. Quindi il ramo più lungo è quello a destra e l'altezza dell'albero è $\log_{\frac{4}{3}} n$. Dato l'Albero di Ricorsione in figura 4 mostriamo che $T(n) = \mathcal{O}(n)$.

$$T(n) \leq cn \cdot \sum_{i=0}^{\log_{\frac{4}{3}} n} (\tfrac{19}{20})^i$$
$$\leq cn \cdot \sum_{i=0}^{\infty} (\tfrac{19}{20})^i = cn \tfrac{1}{(1-\frac{19}{20})} \Rightarrow T(n) = \mathcal{O}(n)$$

Per ottenere il risultato sopra si è sfruttato il fatto che $\sum_{i=0}^{\infty} (\frac{19}{20})^i$ converge a $\frac{1}{(1-\frac{19}{20})}$ poichè è una sommatoria geometrica di ragione $\frac{19}{20} < 1$. (Si veda l'appendice con il promemoria delle sommatorie) Banalmente, $T(n) = \Omega(n)$ poichè $T(n) \geq n$. Se ne conclude che $T(n) = \Theta(n)$.

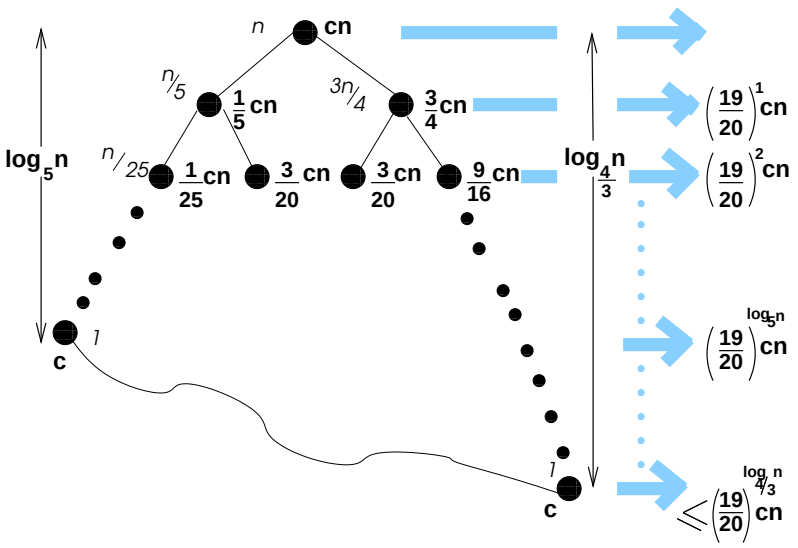

Figure 4: Albero di Ricorsione per $T(n) = T(\frac{n}{5}) + T(\frac{3}{4}n) + \Theta(n)$

2. In Figura 5 vi è l'albero di ricorsione per $T(n) = T(\frac{n}{4}) + T(\frac{3}{4}n) + \Theta(n)$. A destra di ogni livello vi è il costo associato al livello. Il ramo piu' lungo (che determina l'altezza dell'albero) è il ramo a sinistra contenente $k = log_{\frac{4}{3}}n$ archi (perchè "espando il ramo moltiplicando per $\frac{3}{4}$" fino a che $((\frac{3}{4})^k n) = 1$).

   Il ramo piu' corto è invece il ramo piu' a destra contenente $\log_4 n$ nodi.
   Dato l'albero di ricorsione è possibile provare che $T(n) = \mathcal{O}(n \log n)$ e $T(n) = \Omega(n \log n)$.

$$T(n) \leq (\log_{\frac{4}{3}} n + 1) \cdot cn = \Theta(n \log n) \Rightarrow T(n) = \mathcal{O}(n \log n)$$

$$T(n) \geq (\log_4 n + 1) \cdot cn = \Theta(n \log n) \Rightarrow T(n) = \Omega(n \log n)$$

3-c Dato l'Albero di Ricorsione in figura 6 mostriamo che $T(n) = \mathcal{O}(n^2)$.
$T(n) \leq n^2 \cdot \sum_{i=0}^{\log_4 n} (\frac{3}{16})^i$
$\leq n^2 \cdot \sum_{i=0}^{\infty} (\frac{3}{16})^i = n^2 \frac{1}{(1-\frac{3}{16})} \Rightarrow T(n) = \mathcal{O}(n^2)$

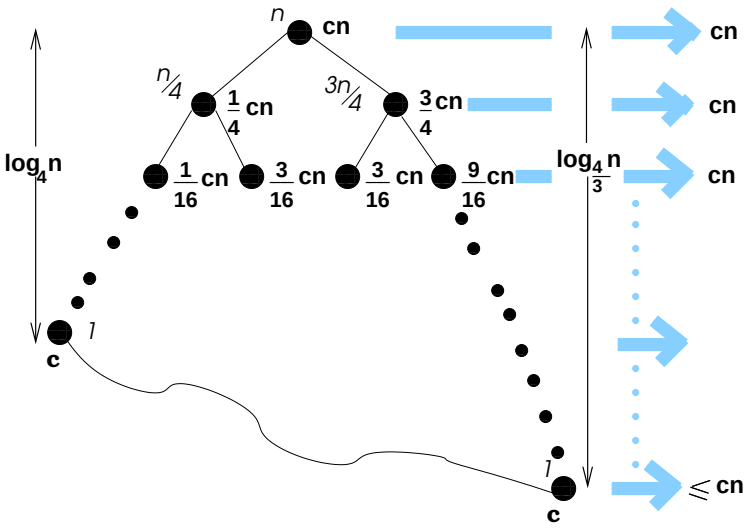

Figure 5: Albero di Ricorsione per $T(n) = T(\frac{n}{4}) + T(\frac{3}{4}n) + \Theta(n)$

Banalmente, $T(n) = \Omega(n^2)$ poichè $T(n) \geq n^2$. Se ne conclude che $T(n) = \Theta(n^2)$.

3-d Questo punto generalizza i punti 3-a e 3-c. L'albero di ricorsione per $T(n) = aT(\frac{n}{b}) + n^p$ sara' un albero completo con:

- altezza $h = log_b(n)$
- numero di foglie $f = a^{log_b n} = n^{log_b a}$ (il costo sulle foglie sara' dunque $\Theta(n^{log_b a})$)
- costo associato ai nodi di profondita' i pari a $a^i(\frac{n}{b^i})^p = (\frac{a}{b^p})^i n^p$

Dato l'albero di ricorsione sopra descritto possiamo scrivere:

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

dove $\Theta(n^{log_b a})$ è il costo sulle foglie ed i termini nella somma sono i costi degli altri livelli. Mostriamo ora che

1. Se $p > log_b a$ allora $T(n) = \Theta(n^p)$.

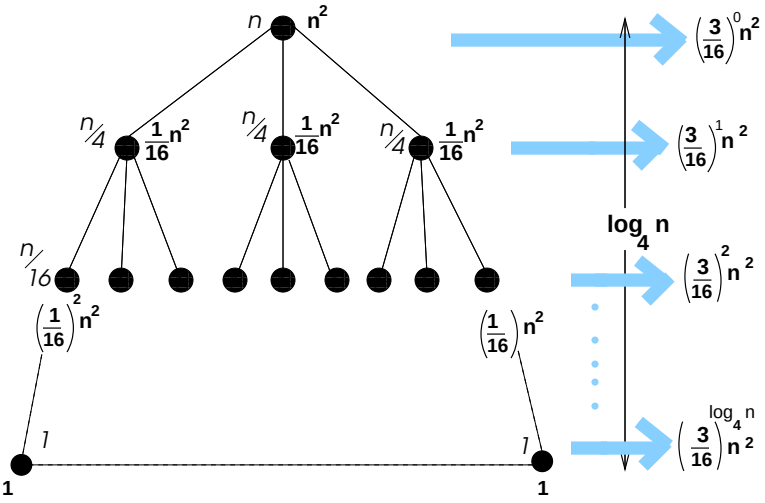

Figure 6: Albero di Ricorsione per $T(n) = 3T(\frac{n}{4}) + n^2$

Se $p > log_b a$, allora $b^p > b^{log_b a} = a$ e la ragione della sommatoria $(\frac{a}{b^p})$ è minore di 1. Dunque $\sum_{i=0}^{\infty}(\frac{a}{b^p})^i$ converge ad una costante c e si ha:
$T(n) = \Theta(n^{log_b a}) + n^p \sum_{i=0}^{log_b n - 1}(\frac{a}{b^p})^i$
$\leq \Theta(n^{log_b a}) + n^p \sum_{i=0}^{\infty}(\frac{a}{b^p})^i =$
$= \Theta(n^{log_b a}) + \Theta(n^p) = \Theta(n^p)$ perchè l'esponente p supera l'esponente $log_b a$!
Si noti che la prima ricorrenza nel punto 3-a e la ricorrenza in $3-c$ ricadevano in questo caso.

2. Se $p = log_b a$ allora $T(n) = \Theta(n^p lg(n))$

   Se $p = log_b a$, allora $b^p = b^{log_b a} = a$ ed ogni livello nell'albero (foglie comprese!) costa n^p. Ci sono $log_b n = \Theta(lg(n))$ livelli nell'albero e dunque:
   $$T(n) = \Theta(n^p(lgn))$$
   Si noti che la seconda ricorrenza nel punto 3-a ricadeva in questo caso.

3. Se $p < log_b a$ allora $T(n) = \Theta(n^{log_b a})$.

   Se $p < log_b a$, allora $b^p < b^{log_b a} = a$ e la ragione della sommatoria $(\frac{a}{b^p})$ è maggiore di 1. Dunque $\sum_{i=0}^{\infty}(\frac{a}{b^p})^i$ non converge ma

possiamo valutare esattamente quanto vale $\sum_{i=0}^{\log_b n-1}(\frac{a}{b^p})^i$ usando la formula $\sum_{i=0}^{p} x^i = \frac{x^{p+1}-1}{x-1}$. Otteniamo:

$$\begin{aligned}
T(n) &= \Theta(n^{log_b a}) + n^p \sum_{i=0}^{log_b n-1}(\tfrac{a}{b^p})^i \\
&= \Theta(n^{log_b a}) + n^p \tfrac{(\frac{a}{b^p})^{log_b n}-1}{\frac{a}{b^p}-1} \\
&= \Theta(n^{log_b a}) + n^p \tfrac{\frac{a^{log_b n}}{b^{log_b n \cdot p}}-1}{\frac{a}{b^p}-1} \\
&= \Theta(n^{log_b a}) + n^p \tfrac{\frac{a^{log_b n}}{n^p}-1}{\frac{a}{b^p}-1} \\
&= \Theta(n^{log_b a}) + \tfrac{a^{log_b n}-\frac{1}{n^p}}{\frac{a}{b^p}-1} \\
&= \Theta(n^{log_b a}) + \tfrac{n^{log_b a}-\frac{1}{n^p}}{\frac{a}{b^p}-1} \\
&= \Theta(n^{log_b a})
\end{aligned}$$

Si noti che la terza ricorrenza nel punto 3-a ricadeva in questo caso.

Uso del cambiamento di variabili: esercizi risolti

Spesso risulta utile/necessario applicare un cambiamento di variabili prima di utilizzare uno dei metodi di soluzione delle ricorrenze visti sopra. Negli esercizi che seguono le ricorrenze vengono "maneggiate" opportunamente mediante il cambio di variabili e quindi risolte con i metodi classici[2].

1. $T(n) = 2T(\sqrt{n}) + \log n$
2. $T(n) = 2T(\frac{n}{2}) + \log n$
3. $T(n) = \sqrt{n}T(\sqrt{n}) + n\log(\sqrt{n})$
4. $T(n) = \sqrt{n}T(\sqrt{n}) + \mathcal{O}(n)$

Soluzione

1. $\boxed{T(n) = 2T(\sqrt{n}) + \log n}$ Poniamo $n = 2^k$ (e $k = \log(n)$). Sostituendo nella nostra ricorrenza otteniamo:

$$T(2^k) = 2T(\sqrt{2^k}) + k = 2T(2^{\frac{k}{2}}) + k$$

[2]In linea di massima il cambio di variabili puo' far assumere un aspetto piu' "appettibile" alle ricorrenze in cui compaiono logaritmi e radici.

Poniamo ora $S(p) = T(2^p)$ e ricaviamo la ricorrenza:

$$S(k) = 2S(\frac{k}{2}) + k$$

Nella ricorrenza $S(k)$ appena scritta riconosciamo la funzione di complessita' di MergeSort. La soluzione di $S(k)$ è dunque $S(k) = \Theta(k(lg(k))$. Riesprimendo la soluzione $S(k) = \Theta(k(lg(k))$ in termini di T ed n otteniamo: $T(2^k) = \Theta(klg(k)) \quad i.e. \quad T(n) = \Theta(lg(n)lg(lg(n)))$

2. $\boxed{T(n) = 2T(\frac{n}{2}) + \log n}$ Poniamo $n = 2^k$ (e $k = \log(n)$). Sostituendo nella nostra ricorrenza otteniamo:

$$T(2^k) = 2T(\frac{2^k}{2}) + k = 2T(2^{k-1}) + k$$

Dividiamo[3] tutto per 2^k e otteniamo:

$$\frac{T(2^k)}{2^k} = 2\frac{T(2^{k-1})}{2^k} + \frac{k}{2^k} = \frac{T(2^{k-1})}{2^{k-1}} + \frac{k}{2^k}$$

Poniamo ora $S(p) = \frac{T(2^p)}{2^p}$ e ricaviamo la ricorrenza:

$$S(k) = S(k-1) + \frac{k}{2^k}$$

La ricorrenza appena scritta si risolve facilmente con lo svolgimento della ricorrenza:

$$\begin{aligned}
S(k) &= S(k-1) + \tfrac{k}{2^k} = \\
&= S(k-2) + \tfrac{k-1}{2^{k-1}} + \tfrac{k}{2^k} = \\
&= \cdots = \\
&= S(1) + \textstyle\sum_{i=1}^{k} i(\frac{1}{2})^i = \\
&= \Theta(1) + \textstyle\sum_{i=1}^{k} i(\frac{1}{2})^i \leq \\
&\leq \Theta(1) + \textstyle\sum_{i=1}^{\infty} i(\frac{1}{2})^i = \\
&= \Theta(1) + \tfrac{1}{1-\frac{1}{2}} = \Theta(1)
\end{aligned}$$

Risostituendo all'indietro otteniamo:

$$\Theta(1) = S(k) = \frac{T(2^k)}{2^k} = \frac{T(n)}{n}$$

[3]per ora vi anticipo che questo serve a "mandare via" il 2 che moltiplica $T(\frac{2^k}{2})\ldots$

$$\frac{T(n)}{n} = \Theta(1) \Rightarrow T(n) = n \cdot \Theta(1) \Rightarrow T(n) = \Theta(n)$$

3. $\boxed{T(n) = \sqrt{n}T(\sqrt{n}) + n\log(\sqrt{n})}$ Al solito, poniamo $n = 2^k$ (e $k = \log(n)$). Sostituendo nella nostra ricorrenza otteniamo:

$$T(2^k) = 2^{\frac{k}{2}}T(2^{\frac{k}{2}}) + 2^k \log(2^{\frac{k}{2}}) = 2^{\frac{k}{2}}T(2^{\frac{k}{2}}) + 2^k \cdot \frac{k}{2}$$

Dividiamo per 2^k e otteniamo:

$$\frac{T(2^k)}{2^k} = \frac{2^{\frac{k}{2}}}{2^k}T(2^{\frac{k}{2}}) + \frac{2^k}{2^k} \cdot \frac{k}{2} = \frac{T(2^{\frac{k}{2}})}{\frac{k}{2}} + \frac{k}{2}$$

Poniamo ora $S(p) = \frac{T(2^p)}{2^p}$ e ricaviamo la ricorrenza:

$$S(k) = S(\frac{k}{2}) + \frac{k}{2}$$

Svolgendo la ricorrenza otteniamo la sommatoria:

$$S(k) = S(1) + k\sum_{i=1}^{\log(k)} \frac{1}{2} \leq \Theta(1) + k\sum_{i=1}^{\infty} \frac{1}{2} = \Theta(1) + k \cdot \frac{1}{1-\frac{1}{2}} = \Theta(k)$$

Risostituendo all'indietro:

$$\frac{T(n)}{n} = \frac{T(2^k)}{2^k} = S(k) = \Theta(k) = \Theta(lg(n)$$

$$\frac{T(n)}{n} = \Theta(lg(n)) \Rightarrow T(n) = \Theta(n \cdot \log(n))$$

4. $\boxed{T(n) = \sqrt{n}T(\sqrt{n}) + \mathcal{O}(n)}$ Esplicitando le costanti nell'$\mathcal{O}$ ho: $T(n) \leq \sqrt{n}T(\sqrt{n}) + cn$ Sostituendo nella nostra ricorrenza otteniamo:

$$T(2^k) \leq 2^{\frac{k}{2}}T(2^{\frac{k}{2}}) + c \cdot 2^k$$

Dividiamo per 2^k e otteniamo:

$$\frac{T(2^k)}{2^k} \leq \frac{2^{\frac{k}{2}}}{2^k}T(2^{\frac{k}{2}}) + c$$

Poniamo ora $S(p) = \frac{T(2^p)}{2^p}$ e ricaviamo la ricorrenza:

$$S(k) \leq S(\frac{k}{2}) + c = S(\frac{k}{2}) + \Theta(1)$$

Sappiamo che $S(k) \leq S(\frac{k}{2}) + \Theta(1)$ sta in $\mathcal{O}(\log(k))$ (si veda il primo punto dell'esercizio 2 o altrimenti si provi a risolvere la ricorrenza con uno dei tre metodi visti: è facilissimo!). Risostituendo all'indietro:

$$\frac{T(n)}{n} = \frac{T(2^k)}{2^k} = S(k) = \mathcal{O}(lg(k)) = \mathcal{O}(lg(lg(n)))$$

$$\frac{T(n)}{n} = \mathcal{O}(lg(lg(n))) \Rightarrow T(n) = \mathcal{O}(n \cdot \log(\log(n)))$$

Esercizi da svolgere

Si analizzino le seguenti ricorrenze dove si assuma $T(n)$ costante per n sufficientemente piccoli:

1. $T(n) = 3T(\frac{n}{3}) + \frac{n}{2}$ [Sol. $T(n) = Theta(n \cdot lg(n))$]
   Si utilizzi il metodo di sostituzione
2. $T(n) = T(n-1) + \mathcal{O}(n^2)$ [Sol. $T(n) = \mathcal{O}(n^3)$]
   Si utilizzi il metodo di iterazione (si tenga presente che $\sum_{i=0}^{n} i^2 = \frac{(n)(n+1)(2n+1)}{6} = \Theta(n^3)$) o il metodo di sostituzione.
3. $T(n) = T(m) + T(n-1-m) + 1$ [Sol. $T(n) = \Theta(n)$]
4. $T(n) = T(m) + T(n-m) + 1$ dove $0 < m < n$ [Sol. $T(n) = \Theta(n)$]
5. In questo esercizio e negli esercizi $4, 5, 6, 7$ è utile fare l'albero di ricorsione. Si consulti gli esercizi risolti sull'albero di ricorsione ed in particolare l'esercizio 3-d. Si disegni sempre l'albero di ricorsione e si facciano bene i conti.
   $T(n) = 4T(\frac{n}{10}) + n$ [Sol. $T(n) = \Theta(n)$]
6. $T(n) = T(\frac{n}{10}) + T(\frac{9n}{10}) + n$ [Sol. $T(n) = \Theta(nlg(n))$]
7. $T(n) \leq 4T(\frac{n}{2}) + n^3$ [Sol. $T(n) = \mathcal{O}(n^3)$]
8. $T(n) = 9T(\frac{n}{2}) + n^3$ [Sol. $T(n) = \Theta(n^{log_2 9})$]

9. $T(n) \leq 4T(\frac{n}{16}) + \sqrt{n}$
   ($\sqrt{n} = n^{\frac{1}{2}}$...anche qui va bene usare albero di ricorsione e, "mimando il caso giusto" dell'esercizio risolto 2-d si avra' $T(n) = \mathcal{O}(\sqrt{n}\log n)$

10. $T(n) = T(\sqrt{n}) + 1$ [Sol. $T(n) = \Theta(lg(lg(n))$]
    (si usi il cambio di variabili...)

11. $T(n) = T(\frac{n}{2}) + lg^2(n)$ [Sol. $T(n) = \Theta(lg^3 n)$]
    Anche qui è molto utile il cambio di variabili (si tenga presente inoltre che $\sum_{i=0}^{n} i^2 = \frac{(n)(n+1)(2n+1)}{6} = \Theta(n^3)$).

12. $T(n) = 2T(\frac{n}{2}) + n \cdot lg^2(n)$ [Sol. $T(n) = \Theta(n \cdot lg^3 n)$]