Corso di laurea in Informatica
Introduzione agli Algoritmi
Didattica blended

Esercizi sulle equazioni di ricorrenza

Angelo Monti

SAPIENZA
UNIVERSITÀ DI ROMA

Sulla base delle slides a cura di T. Calamoneri e G. Bongiovanni per il corso di informatica generale AA 2019/2020

- **Risolvere la seguente equazione con i quattro metodi visti a lezione ove possibile:**

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

Metodo principale

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

$a = 2,\ b = 2,$ e $f(n) = \Theta(n \log n)$
$n^{\log_b a} = n^{\log_2 2} = n$

Ora, $f(n) = \Theta(n \log n)$ è asintoticamente più grande di $n^{\log_b a} = n$, ma non polinomialmente più grande. Infatti, $\log n$ è asintoticamente minore di n^{ϵ} per qualunque valore di $\epsilon > 0$.
In pratica: $f(n) = \Omega(n^{\log_a b})$ ma $f(n) \neq \Omega(n^{\log_a b - \epsilon})$ qualunque sia la costante $\epsilon > 0$.
Di conseguenza **non possiamo applicare il metodo del teorema principale**.

- **1) Risolvere la seguente equazione di ricorrenza col metodo iterativo**
- $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n \log n)$
- $T(1) = \Theta(1)$

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

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

$$= 2\left(2\left(2T\left(\frac{n}{2^3}\right) + \Theta\left(\frac{n}{4}\log\frac{n}{4}\right)\right) + \Theta\left(\frac{n}{2}\log\frac{n}{2}\right)\right) + \Theta(n \log n)$$

.....

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

ESERCIZIO:

segue metodo iterativo per $T(n) = 2T(n/2) + \Theta(n \log n)$ e $T(1) = \Theta(1)$

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

…….. Ci fermiamo quando $k = \log n$ e otteniamo:

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

$$= n\Theta(1) + \Theta\left(n \sum_{i=0}^{\log n - 1} \log n + n \sum_{i=0}^{\log n - 1} \log 2^i\right)$$

$$= \Theta(n) + \Theta\left(n \log^2 n - n \sum_{i=0}^{\log n - 1} i\right)$$

$$= \Theta(n) + \Theta\left(n \log^2 n - n \frac{\log n(\log n - 1)}{2}\right)$$

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

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

- **2) Risolvere la seguente equazione di ricorrenza col metodo di sostituzione:**
- $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n \log n)$
- $T(1) = \Theta(1)$

Impostiamo la dimensione del caso base a 2, per evitare di dover gestire il caso di $\log 1 = 0 \Longrightarrow T(2) = \Theta(1)$.

Dobbiamo innanzi tutto eliminare la notazione asintotica, quindi l'equazione diventa:

$$T(n) = 2T\left(\frac{n}{2}\right) + a \cdot n \log n$$ per qualche costante a > 0

$$T(2) = b$$ per qualche costante b > 0

ESERCIZIO:

segue metodo di sostituzione per $T(n) = 2T(n/2) + \Theta(n \log n)$ *e* $T(2) = \Theta(1)$

Proviamo a dimostrare per induzione che la soluzione è:

$T(n) \leq c \cdot n \log^2 n$, dove *c* è una costante da determinare.

Passo base. $T(2) = b \leq c \cdot 2 \cdot 1$, vera per $c \geq \frac{b}{2}$.

Passo induttivo. Sostituendo nell'equazione generica otteniamo:

$$\begin{aligned} T(n) &= 2c\frac{n}{2}\log^2\frac{n}{2} + a \cdot n \log n \\ &= c \cdot (\log n - 1)^2 + a \cdot n \log n \\ &= c \cdot n(\log^2 n - 2\log n + 1) + a \cdot n \log n \\ &= c \cdot n \log^2 n - 2 \cdot c \cdot n \log n + c \cdot n + a \cdot n \log n \\ &\leq c \cdot n \log^2 n - c \cdot n \log n + a \cdot n \log n \quad \text{(perché } c \cdot n \log n \geq c \cdot n\text{)} \\ &\leq c \cdot n \log^2 n \quad \textit{che è vera per } c \geq a \end{aligned}$$

Concludiamo che $T(n) = O(n \log^2)$

Si lascia come esercizio *provare che* $T(n) = \Omega(n \log^2 n)$

ESERCIZIO:

segue metodo di sostituzione per $T(n) = 2T(n/2) + \Theta(n \log n)$ *e* $T(2) = \Theta(1)$

Proviamo a dimostrare per induzione che la soluzione è:

$T(n) \leq c \cdot n \log^2 n$, dove *c* è una costante da determinare.

Passo base. $T(2) = b \leq c \cdot 2 \cdot 1$, vera per $c \geq \frac{b}{2}$.

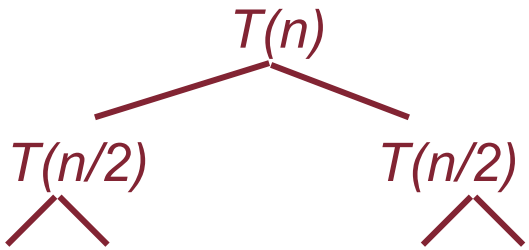
Passo induttivo. Sostituendo nell'equazione generica otteniamo:

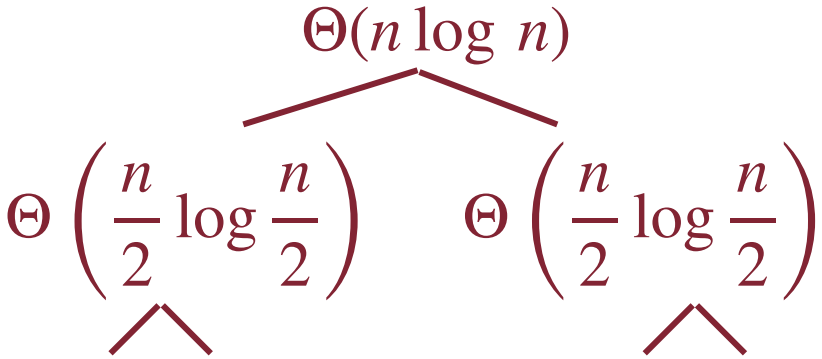
$$
\begin{aligned}
T(n) &= 2c\frac{n}{2}\log^2\frac{n}{2} + a \cdot n \log n \\
&= c \cdot (\log n - 1)^2 + a \cdot n \log n \\
&= c \cdot n(\log^2 n - 2\log n + 1) + a \cdot n \log n \\
&= c \cdot n \log^2 n - 2 \cdot c \cdot n \log n + c \cdot n + a \cdot n \log n \\
&\leq c \cdot n \log^2 n - c \cdot n \log n + a \cdot n \log n \quad \text{(perché } c \cdot n \log n \geq c \cdot n\text{)} \\
&\leq c \cdot n \log^2 n \quad \textit{che è vera per } c \geq a
\end{aligned}
$$

Concludiamo che $T(n) = O(n \log^2)$

Si lascia come esercizio *provare che* $T(n) = \Omega(n \log^2 n)$

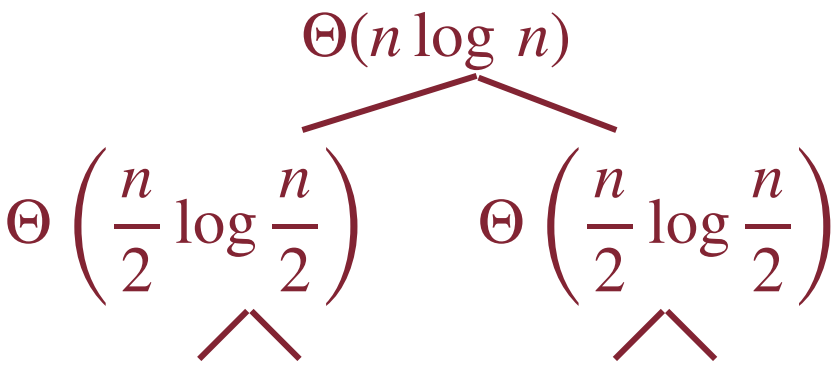
- **3) Risolvere la seguente equazione di ricorrenza col metodo dell'albero**
- $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n \log n)$
- $T(1) = \Theta(1)$





ESERCIZIO:

segue metodo dell'albero per $T(n) = 2T(n/2) + \Theta(n \log n)$ e $T(1) = \Theta(1)$



#nodi	contributo
2^0	$\Theta(n \log n)$
2^1	$\Theta\left(\frac{n}{2}\log\frac{n}{2}\right)$
…	…
2^i.	$\Theta\left(\frac{n}{2^i}\log\frac{n}{2^i}\right)$
…	…

finché $n/2^i = 1$ cioè $i = \log n$

Sommo tutti i contributi:

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

Corso di laurea in Informatica
Introduzione agli Algoritmi
Didattica blended

Esercizi per casa

SAPIENZA
UNIVERSITÀ DI ROMA

Esercizio:

- Calcolare l' equazione di ricorrenza associata al seguente algoritmo ricorsivo e risolverla con tutti i metodi possibili.
- L'algoritmo è invocato con $a = 0 \quad e \quad b = n - 1$ dove n è la lunghezza della lista:

```
def Palindromo(lista, a,b)
     if b ≤ a :
          return True
     if lista[a]!= lista[b]:
          return False
     return Palindromo(lista, a+1, b-1)
```

Esercizio:

- Calcolare l' equazione di ricorrenza associata al seguente algoritmo e risolverla con tutti i metodi possibili:

```
def Test(n):
    k = 0
    for i in range(1,n+1):
        k += 1
    if n ≤ 1:
        return k
    return Test(n// 2)+ Test(n // 4)
```