

# La Ricorsione e La Complessità

## Informatica

Alberto Casagrande  
*Email:* `acasagrande@units.it`

a.a. 2019/2020

# Algoritmi Ricorsivi e Complessità

# Algoritmi Ricorsivi

Fino a ora abbiamo visto **algoritmi iterativi**

Possiamo calcolarne la complessità contando il numero di iterazioni.

# Algoritmi Ricorsivi

Fino a ora abbiamo visto **algoritmi iterativi**

Possiamo calcolarne la complessità contando il numero di iterazioni.

Possiamo sfruttare la ricorsione scrivendo gli algoritmi?

# Algoritmi Ricorsivi

Fino a ora abbiamo visto **algoritmi iterativi**

Possiamo calcolarne la complessità contando il numero di iterazioni.

Possiamo sfruttare la ricorsione scrivendo gli algoritmi?

Come calcolarne la complessità?

# Un Algoritmo Ricorsivo ... Familiare

```
def do_something(A, r=|A|):  
    if r>1  
        j ← i  
        while (j>1 and A[j]<A[j-1]):  
            swap(A, j-1, j)  
            j←j-1  
        endwhile  
        do_something(A, r-1)  
    endif  
enddef
```

Non è familiare questo codice? È **insertion sort** ricorsivo!

# Come Calcolarne la Complessità?

Usando le **equazioni ricorsive di complessità!!!**

Se  $T(|A|)$  il tempo per risolvere il problema su  $A$  allora

# Come Calcolarne la Complessità?

Usando le **equazioni ricorsive di complessità!!!**

Se  $T(|A|)$  il tempo per risolvere il problema su  $A$  allora

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

Per induzione,  $T(|A|) = \sum_{i=0}^{|A|} \Theta(i) = \Theta(|A|^2)$



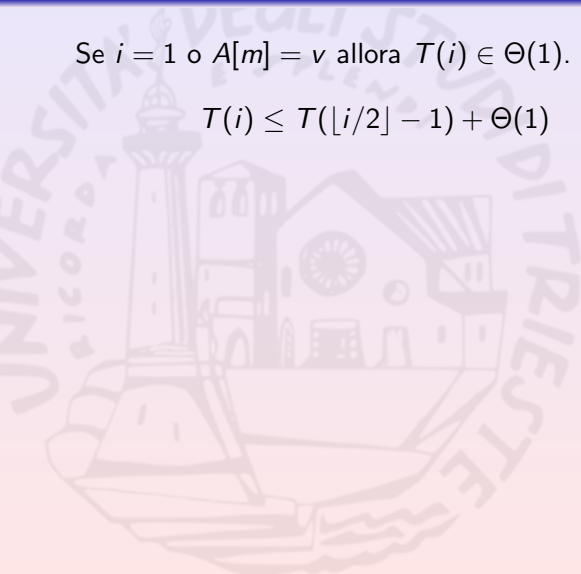
## Un Esempio Più Spinoso

```
def rec_find(A,v,l=1,r=|A|):  
    if r < l  
        return 0  
    endif  
  
    m ← (l+r)/2  
    if A[m]=v  
        return m  
    endif  
    if A[m]>v  
        return rec_find(A, v, l, m-1)  
    else  
        return rec_find(A, v, m+1, r)  
    endif  
enddef
```

# Scriviamo l'Equazione Ricorsiva...

Se  $i = 1$  o  $A[m] = v$  allora  $T(i) \in \Theta(1)$ . Altrimenti

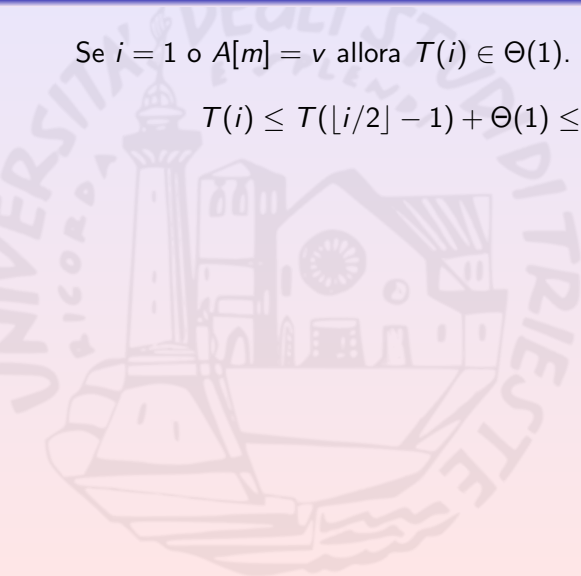
$$T(i) \leq T(\lfloor i/2 \rfloor - 1) + \Theta(1)$$



## Scriviamo l'Equazione Ricorsiva...

Se  $i = 1$  o  $A[m] = v$  allora  $T(i) \in \Theta(1)$ . Altrimenti

$$T(i) \leq T(\lfloor i/2 \rfloor - 1) + \Theta(1) \leq T(i/2) + \Theta(1)$$



# Scriviamo l'Equazione Ricorsiva...

Se  $i = 1$  o  $A[m] = v$  allora  $T(i) \in \Theta(1)$ . Altrimenti

$$T(i) \leq T(\lfloor i/2 \rfloor - 1) + \Theta(1) \leq T(i/2) + \Theta(1)$$

Poniamo  $j \stackrel{\text{def}}{=} \log_2 i$  e  $P(j) \stackrel{\text{def}}{=} T(2^j)$

# Scriviamo l'Equazione Ricorsiva...

Se  $i = 1$  o  $A[m] = v$  allora  $T(i) \in \Theta(1)$ . Altrimenti

$$T(i) \leq T(\lfloor i/2 \rfloor - 1) + \Theta(1) \leq T(i/2) + \Theta(1)$$

Poniamo  $j \stackrel{\text{def}}{=} \log_2 i$  e  $P(j) \stackrel{\text{def}}{=} T(2^j)$

$$\begin{aligned} P(j) &= T(2^j) \\ &\leq T(2^j/2) + \Theta(1) \\ &= T(2^{j-1}) + \Theta(1) = P(j-1) + \Theta(1) \end{aligned}$$

Quindi  $P(j) \leq \sum_{i=0}^j \Theta(1) = \Theta(j)$ ,  $P(j) = O(j)$  e ...

$$T(|A|) = P(\log_2 |A|) = O(\log |A|)$$

# Quando l'Equazione è Più Complicata?

Per esempio:

$$T(i) = \begin{cases} \Theta(1) & \text{se } i = 1 \\ 2 * T(i/2) + \Theta(i) & \text{se } i > 1 \end{cases}$$

Tre metodi:

- metodo di sostituzione
- l'albero di ricorsione
- il teorema dell'esperto

## Metodo di Sostituzione

## Il Metodo di Sostituzione

Due passi:

- 1 ipotizzo una complessità per  $T(n)$
- 2 dimostro per induzione che esistono delle costanti che soddisfano l'ipotesi



## Il Metodo di Sostituzione: Un Esempio

Es.  $T(n) = 2 * T(n/2) + O(n)$

- 1 ipotizzo che  $T(n) \in O(n \log n)$
- 2 assumo che  $\exists c \forall m < n \ T(m) \leq c * m * \log_2 m$

$$\begin{aligned} T(n) &= 2 * T(n/2) + c' * n \\ &= 2 * (c * n/2 * \log_2(n/2)) + c' * n \\ &= c * n * \log_2(n/2) + c' * n \\ &= c * n * \log_2(n) - c * n * \log_2(2) + c' * n \end{aligned}$$

Se scelgo  $c' < c$ , allora  $T(n) \leq c * n * \log_2(n)$ .

## Il Metodo di Sostituzione: Un Esempio

Es.  $T(n) = 2 * T(n/2) + O(n)$

- 1 ipotizzo che  $T(n) \in O(n \log n)$
- 2 assumo che  $\exists c \forall m < n \ T(m) \leq c * m * \log_2 m$

$$\begin{aligned} T(n) &= 2 * T(n/2) + c' * n \\ &= 2 * (c * n/2 * \log_2(n/2)) + c' * n \\ &= c * n * \log_2(n/2) + c' * n \\ &= c * n * \log_2(n) - c * n * \log_2(2) + c' * n \end{aligned}$$

Se scelgo  $c' < c$ , allora  $T(n) \leq c * n * \log_2(n)$ .

La costante  $c$  è la stessa per  $n$

## Il Metodo di Sostituzione: Come **NON** fare

Es.  $T(n) = 3 * T(n/2) + O(n)$

- ① ipotizzo che  $T(n) \in O(n * \log n)$
- ② assumo che  $\exists c \forall m < n \ T(m) \leq c * m * \log_2 m$

$$= 3 * (c * n/2 * \log_2(n/2)) + c' * n$$

$$= \frac{3}{2} * c * n * \log_2(n/2) + c' * n$$

$$= \frac{3}{2} * c * n * \log_2(n) - c * n * \log_2(2) + c' * n$$

$$\leq c * \frac{3}{2} * n * \log_2(n)$$

# Il Metodo di Sostituzione: Come **NON** fare

Es.  $T(n) = 3 * T(n/2) + O(n)$

- ① ipotizzo che  $T(n) \in O(n * \log n)$
- ② assumo che  $\exists c \forall m < n \ T(m) \leq c * m * \log_2 m$

$$= 3 * (c * n/2 * \log_2(n/2)) + c' * n$$

$$= \frac{3}{2} * c * n * \log_2(n/2) + c' * n$$

$$= \frac{3}{2} * c * n * \log_2(n) - c * n * \log_2(2) + c' * n$$

$$\leq c * \frac{3}{2} * n * \log_2(n) \in O(n \log n)$$

# Il Metodo di Sostituzione: Come **NON** fare

Es.  $T(n) = 3 * T(n/2) + O(n)$

- ① ipotizzo che  $T(n) \in O(n * \log n)$
- ② assumo che  $\exists c \forall m < n \ T(m) \leq c * m * \log_2 m$

$$= 3 * (c * n/2 * \log_2(n/2)) + c' * n$$

$$= \frac{3}{2} * c * n * \log_2(n/2) + c' * n$$

$$= \frac{3}{2} * c * n * \log_2(n) - c * n * \log_2(2) + c' * n$$

$$\leq c * \frac{3}{2} * n * \log_2(n) \notin O(n \log n)$$

La costante per  $m < n$  era  $c$ , mentre per  $n$  è  $\frac{3}{2} * c$

## Albero di Ricorsione

# Albero di Ricorsione

Costruisci un albero (grafo connesso aciclico) in cui ogni nodo:

- rappresenta una chiamata ricorsiva
- corrisponde al costo della chiamata senza i passi ricorsivi
- è collegato alle chiamate ricorsive che genera

Sommando il costo di tutti i nodi, otteniamo il costo totale.

# Albero di Ricorsione: Un Esempio

Consideriamo l'equazione

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 8 * T(n/2) + \Theta(n^2) & \text{se } n > 1 \end{cases}$$



# Albero di Ricorsione: Un Esempio

Consideriamo l'equazione

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 8 * T(n/2) + \Theta(n^2) & \text{se } n > 1 \end{cases}$$

Scegliamo una funzione in  $\Theta(n^2)$  (es.  $c * n$ )

# Albero di Ricorsione: Un Esempio

Consideriamo l'equazione

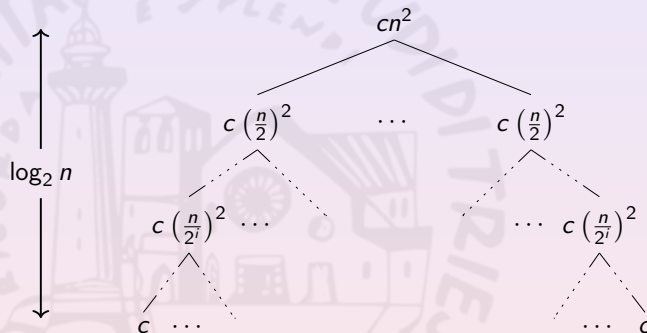
$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 8 * T(n/2) + \Theta(n^2) & \text{se } n > 1 \end{cases}$$

Scegliamo una funzione in  $\Theta(n^2)$  (es.  $c * n$ )

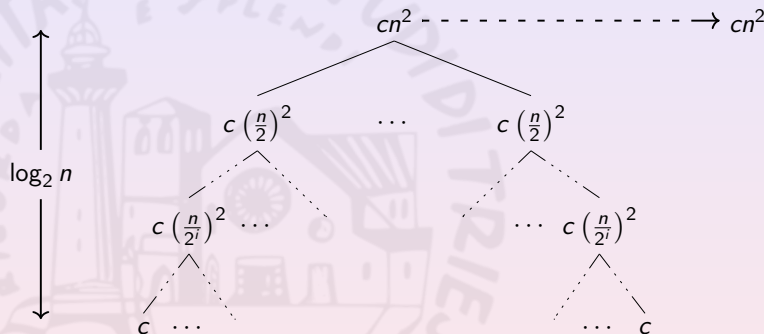
Se  $m$  è la dimensione dell'input per una chiamata:

- il costo su una chiamata è  $cm^2$
- ogni nodo dell'albero ha 8 figli (le chiamate ricorsive)
- $m$  si dimezza a ogni chiamata ricorsiva

## Albero di Ricorsione: Un Esempio (Cont'd)

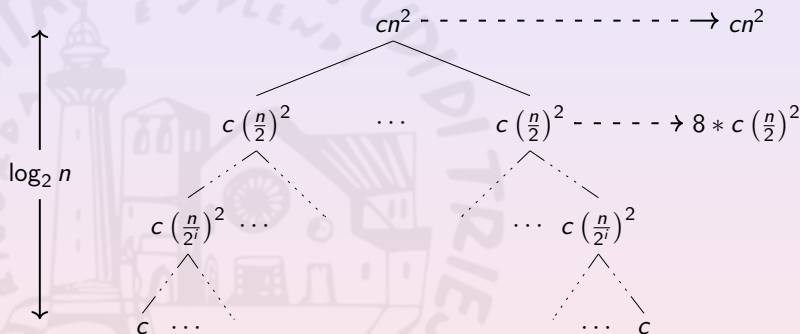


## Albero di Ricorsione: Un Esempio (Cont'd)



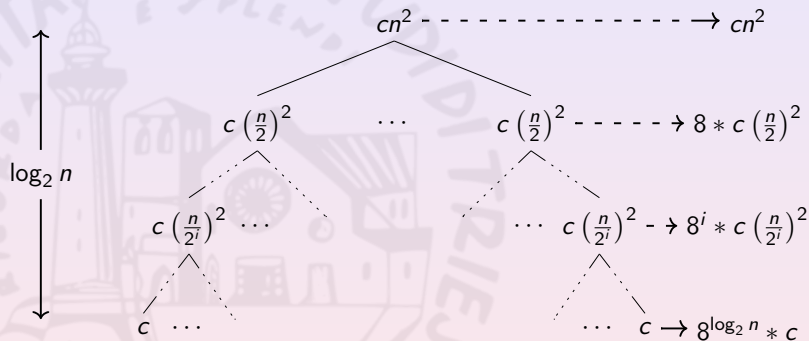
$$T_M(n) = cn^2 \left( 1 + \right)$$

## Albero di Ricorsione: Un Esempio (Cont'd)



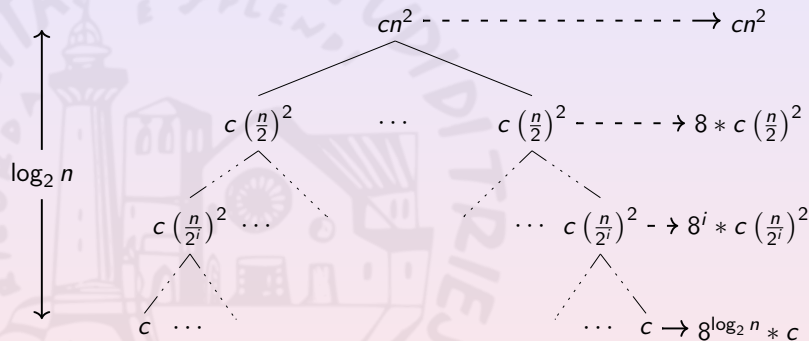
$$T_M(n) = cn^2 \left( 1 + 2 + \dots + 2^{\log_2 n} \right)$$

# Albero di Ricorsione: Un Esempio (Cont'd)



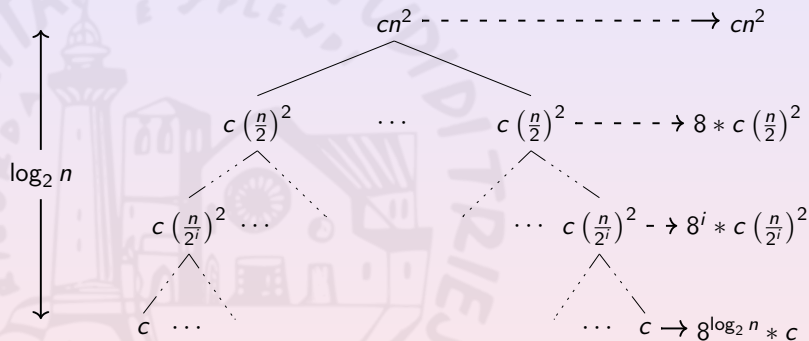
$$T_M(n) = cn^2 \left(1 + 2 + \dots + 2^i + \dots + 2^{\log_2 n}\right)$$

# Albero di Ricorsione: Un Esempio (Cont'd)



$$\begin{aligned}
 T_M(n) &= cn^2 \left( 1 + 2 + \dots + 2^i + \dots + 2^{\log_2 n} \right) \\
 &= cn^2 \left( 2^{1+\log_2 n} - 1 \right) = cn^2 (2n - 1)
 \end{aligned}$$

# Albero di Ricorsione: Un Esempio (Cont'd)



$$\begin{aligned} T_M(n) &= cn^2 \left( 1 + 2 + \dots + 2^i + \dots + 2^{\log_2 n} \right) \\ &= cn^2 \left( 2^{1+\log_2 n} - 1 \right) = cn^2 (2n - 1) \in \Theta(n^3) \end{aligned}$$



## Il Metodo dell'Esperto

# Metodo dell'Esperto

## Theorem (Teorema dell'Esperto)

*Siano  $a \geq 1$  e  $b > 1$  due costanti,  $f(n)$  una funzione  $\geq 0$  e*

$$T(n) \stackrel{\text{def}}{=} \begin{cases} \Theta(1) & \text{se } n = 1 \\ a * T(n/b) + f(n) & \text{altrimenti} \end{cases}$$

*Se, per qualche costante  $\epsilon$ ,  $f(n)$  sta in*

- ❶  $O(n^{\log_b a - \epsilon})$ , allora  $T(n) \in \Theta(n^{\log_b a})$ ;
- ❷  $\Theta(n^{\log_b a})$ , allora  $T(n) \in \Theta(n^{\log_b a} * \log n)$ ;
- ❸  $\Omega(n^{\log_b a + \epsilon})$  e  $a * f(n/b) \leq c * f(n)$  per qualche  $c < 1$  e  $n$  sufficientemente grande, allora  $T(n) \in \Theta(f(n))$ .

# Applicazioni

Consideriamo:

$T(n) = 9 * T(n/3) + O(n)$ :  $a = 9$ ,  $b = 3$  e  $f(n) \in O(n^{\log_3 9 - \epsilon})$   
con  $\epsilon = 1$  quindi **Caso 1**  $\Rightarrow T(n) \in \Theta(n^2)$

# Applicazioni

Consideriamo:

$T(n) = 9 * T(n/3) + O(n)$ :  $a = 9$ ,  $b = 3$  e  $f(n) \in O(n^{\log_3 9 - \epsilon})$   
con  $\epsilon = 1$  quindi **Caso 1**  $\Rightarrow T(n) \in \Theta(n^2)$

$T(n) = T(2 * n/3) + 1$ :  $a = 1$ ,  $b = 3/2$  e  
 $f(n) \in \Theta(n^{\log_{3/2} 1}) = \Theta(1)$  quindi **Caso 2**  $\Rightarrow T(n) \in \Theta(\log n)$

# Applicazioni

Consideriamo:

$T(n) = 9 * T(n/3) + O(n)$ :  $a = 9$ ,  $b = 3$  e  $f(n) \in O(n^{\log_3 9 - \epsilon})$   
con  $\epsilon = 1$  quindi **Caso 1**  $\Rightarrow T(n) \in \Theta(n^2)$

$T(n) = T(2 * n/3) + 1$ :  $a = 1$ ,  $b = 3/2$  e  
 $f(n) \in \Theta(n^{\log_{3/2} 1}) = \Theta(1)$  quindi **Caso 2**  $\Rightarrow T(n) \in \Theta(\log n)$

$T(n) = 3 * T(n/4) + n * \log n$ :  $a = 3$ ,  $b = 4$  e  $f(n) \in \Omega(n^{\log_4 3 + \epsilon})$   
con  $\epsilon \approx 0.2$  quindi **Caso 3**  $\Rightarrow T(n) \in \Theta(n * \log n)$

# Applicazioni

Consideriamo:

$$T(n) = 2 * T(n/2) + n * \log n: a = 2, b = 2 \text{ e } f(n) \in \Omega(n^{\log_2 2}).$$

Sfortunatamente,

$$\lim_{n \rightarrow \infty} \frac{n * \log n}{n} < n^\epsilon$$

per ogni  $\epsilon > 0$  e **NON** si può applicare il teorema dell'esperto.

# Dimostrazione

## Lemma

*Siano  $a \geq 1$  e  $b > 1$  due costanti,  $f(n)$  una funzione  $\geq 0$  e*

$$T(n) \stackrel{\text{def}}{=} \begin{cases} \Theta(1) & \text{se } n = 1 \\ a * T(n/b) + f(n) & \text{se } n = b^i \end{cases}$$

*dove  $i$  è un naturale positivo. Allora*

$$T(n) \in \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

Dim. tramite l'albero di ricorsione.

# Dimostrazione (Cont'd)

## Lemma

Siano  $a \geq 1$  e  $b > 1$  due costanti,  $f(n)$  una funzione  $\geq 0$  e:

$$g(n) \stackrel{\text{def}}{=} \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

- ❶ se  $f(n) \in O(n^{\log_b a - \epsilon})$ , allora  $g(n) \in O(n^{\log_b a})$ ;
- ❷ se  $f(n) \in \Theta(n^{\log_b a})$ , allora  $g(n) \in \Theta(n^{\log_b a} \log n)$ ;
- ❸ se  $a * f(n/b) \leq cf(n)$  per qualche  $c < 1$  e  $n$  sufficientemente grande, allora  $g(n) \in \Theta(f(n))$ .

Dim. tramite sostituzione delle condizioni