

Algoritmi e Strutture Dati

**Valutazione del tempo di esecuzione
degli algoritmi**

Stima del limite asintotico superiore

- Nelle prossimi lucidi definiremo un semplice metodo per ***stimare il limite asintotico superiore $O(.)$*** del tempo di esecuzione di ***algoritmo iterativi***.
 - Stabilire il limite superiore per le operazioni elementari
 - Stabilire il limite superiore per le strutture di controllo
- Ci da un limite superiore che funge da stima, ***non garantisce*** di trovare la ***funzione precisa del tempo di esecuzione***.

Tempo di esecuzione: operazioni semplici

Operazioni Semplici

- *operazioni aritmetiche* (+, *, ...)
- *operazioni logiche* (&&, ||, ...)
- *confronti* (\leq , \geq , =, ...)
- *assegnamenti* (a = b) senza chiamate di funzione
- *operazioni di lettura* (read)
- *operazioni di controllo* (break, continue, return)

$$T(n) = \Theta(1) \Rightarrow T(n) = O(1)$$

Tempo di esecuzione: ciclo for

Ciclo-for

inizializza $O(1)$

$O(1)$

test

$g(n)$
volte

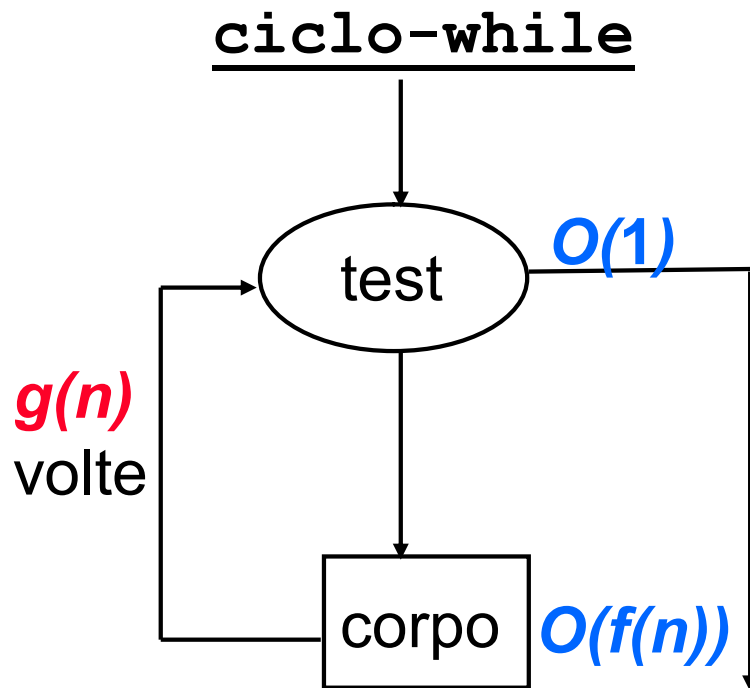
corpo $O(f(n))$

reinizializza $O(1)$

stabilire $g(n)$ è in genere semplice.

$$T(n) = O(g(n) \times f(n))$$

Tempo di esecuzione: ciclo while



Bisogna stabilire un limite per il numero di iterazioni del ciclo, $g(n)$.

Può essere necessaria una prova induttiva per $g(n)$.

$$T(n) = O(g(n) \times f(n))$$

Ciclo while: esempio

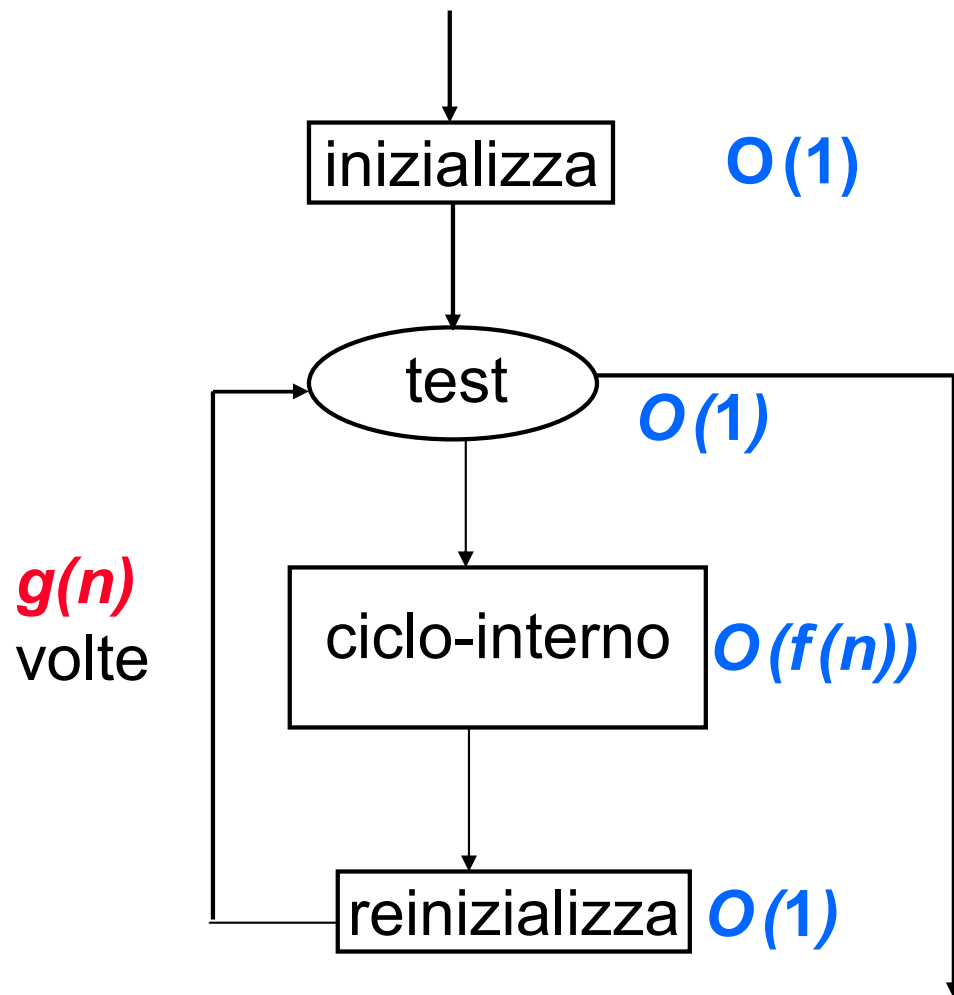
Ricerca dell'elemento x all'interno di un array $A[1...n]$:

```
i = 0           (1)
while x ≠ A[i] (2)
    i = i+1     (3)
```

(1)	$O(1)$
test in (2)	$O(1)$
(3)	$O(1)$
iterazioni	massimo n

$O(\text{ciclo-while}) = O(1) + n O(1) = O(n)$

Tempo di esecuzione: cicli innestati



$$T(n) = O(g(n) \times f(n))$$

Cicli innestati: esempio

```
for i = 1 to n
  for j = 1 to n
    k = i + j
```

$\left. \begin{array}{l} \left. \begin{array}{l} \left. \begin{array}{l} \text{for } i = 1 \text{ to } n \\ \text{for } j = 1 \text{ to } n \\ k = i + j \end{array} \right\} = O(n) \end{array} \right\} = O(n^2) \end{array} \right\} = O(n^2)$

$$T(n) = O(n \times n) = O(n^2)$$

Cicli innestati: esempio

```
for i = 1 to n
```

```
  for j = i to n
```

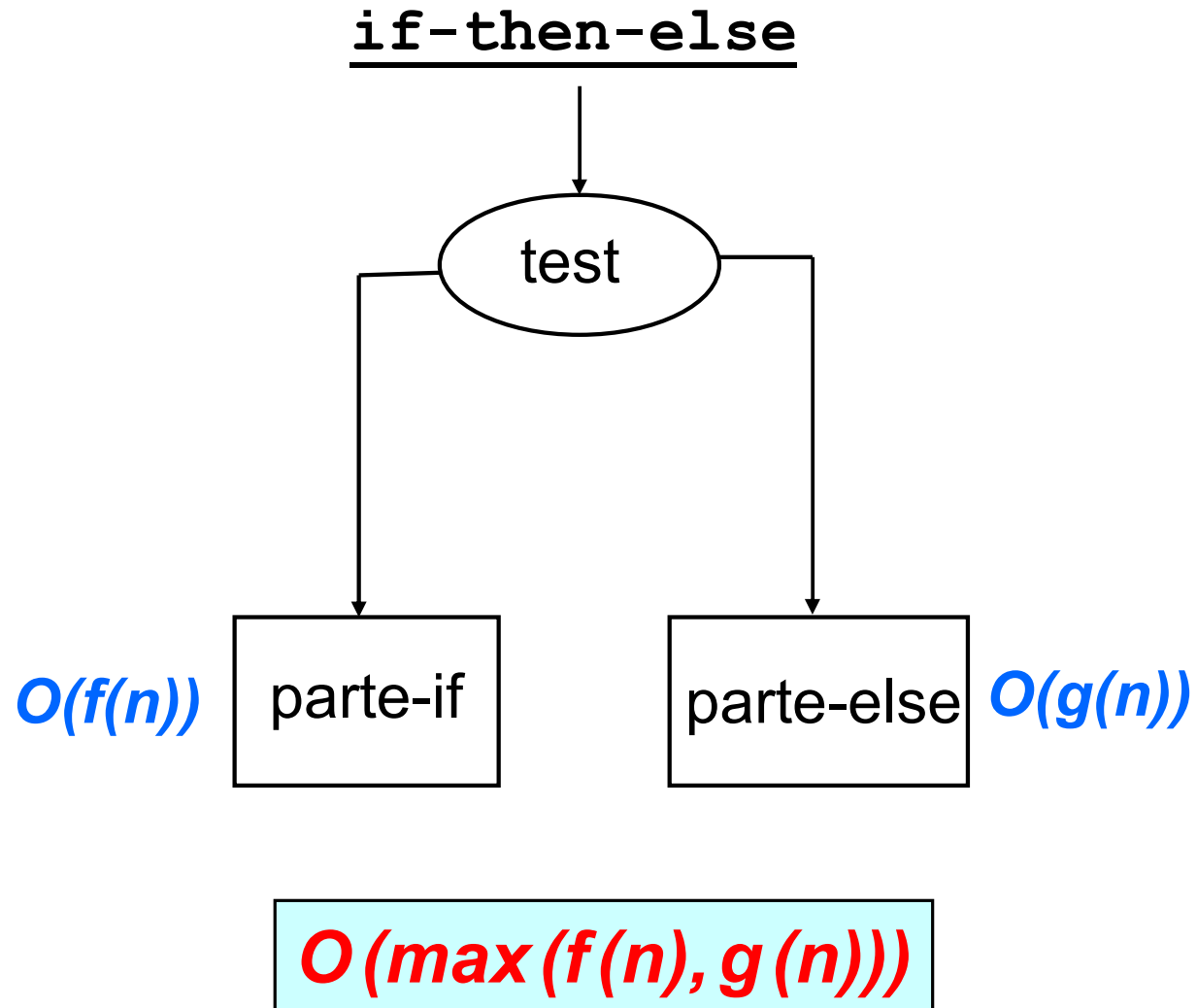
```
    k = i + j
```

} = $O(n-i)$

} = $O(n^2)$

$$T(n) = O(n \times n) = O(n^2)$$

Tempo di esecuzione: If-Then-Else



If-Then-Else: esempio

```
if A[1][i] = 0 then
  for i = 1 to n
    for j = 1 to n
      a[i][j] = 0
else
  for i = 1 to n
    A[i][i] = 1
```

$\left. \begin{array}{l} \text{for } j = 1 \text{ to } n \\ a[i][j] = 0 \end{array} \right\} = O(n)$

$\left. \begin{array}{l} \text{for } i = 1 \text{ to } n \\ A[i][i] = 1 \end{array} \right\} = O(n)$

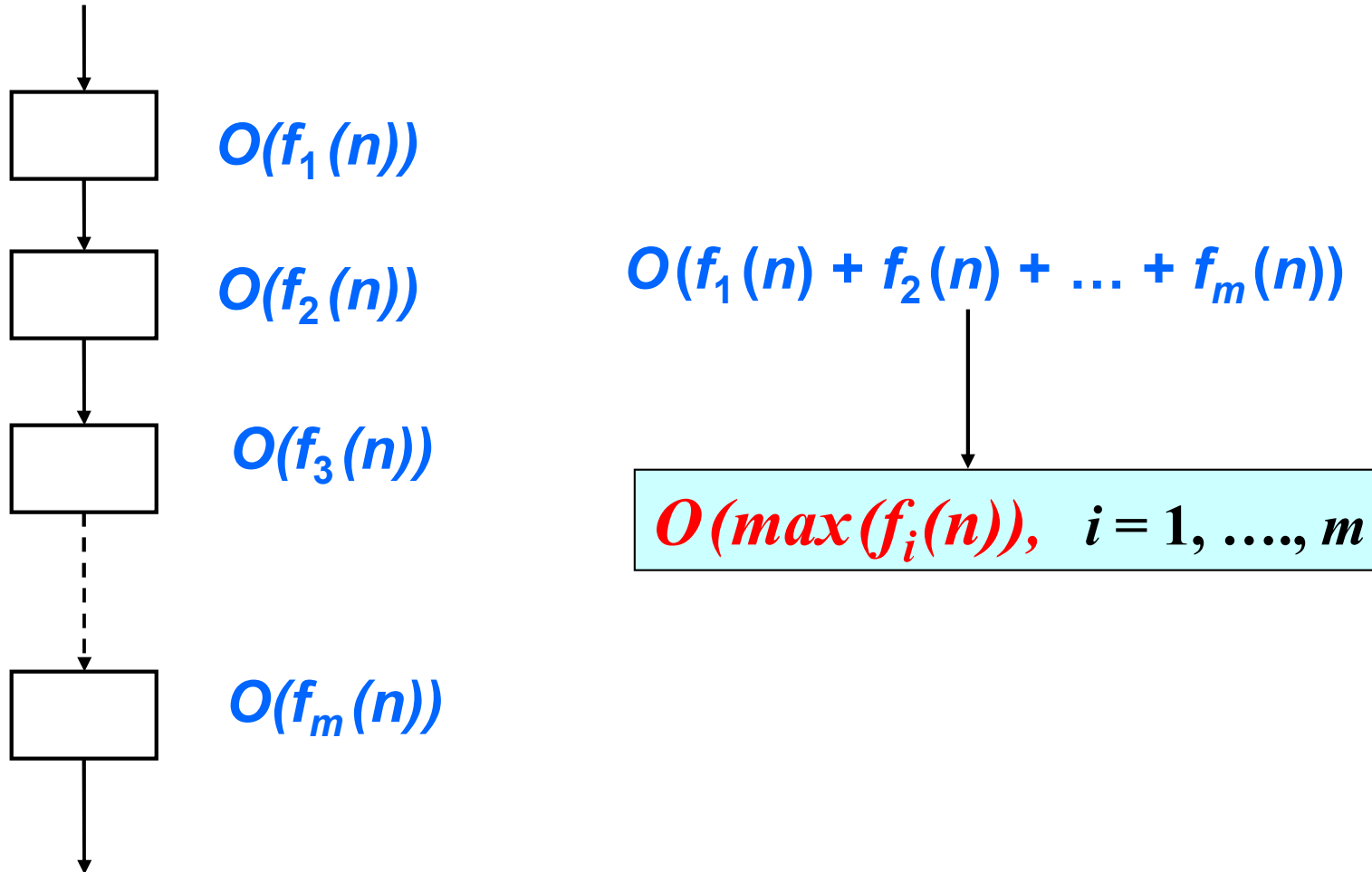
$\left. \begin{array}{l} \left. \begin{array}{l} \text{for } j = 1 \text{ to } n \\ a[i][j] = 0 \end{array} \right\} = O(n) \\ \left. \begin{array}{l} \text{for } i = 1 \text{ to } n \\ A[i][i] = 1 \end{array} \right\} = O(n) \end{array} \right\} = O(n^2)$

if: $T(n) = O(n^2)$

else : $T(n) = O(n)$

$$T(n) = \max(O(n^2), O(n)) = O(n^2)$$

Tempo di esecuzione: blocchi sequenziali



Blocchi sequenziali: esempio

```
for i = 1 to n  
  A[1] = 0  
for i = 1 to n  
  for j = 1 to n  
    A[i] = A[i] + A[i]
```

$$\left. \begin{array}{l} \text{for } i = 1 \text{ to } n \\ \quad A[1] = 0 \end{array} \right\} = O(n)$$
$$\left. \begin{array}{l} \text{for } i = 1 \text{ to } n \\ \quad \text{for } j = 1 \text{ to } n \\ \quad \quad A[i] = A[i] + A[i] \end{array} \right\} = O(n) \left. \right\} = O(n^2)$$

$$\begin{aligned} T(n) &= O(\max(f(\text{ciclo-1}), f(\text{ciclo-2}))) \\ &= O(\max(n, n^2)) \\ &= O(n^2) \end{aligned}$$

Esempio: Insert Sort

$$O(n^2) = \left\{ \begin{array}{l} \text{InsertSort(array A[1..n])} \\ \text{for } j = 2 \text{ to } n \\ \quad \text{key} = A[j] \qquad \qquad \qquad = O(1) \\ \quad i = j - 1 \qquad \qquad \qquad = O(1) \\ \quad \text{while } i > 0 \text{ and } A[i] > \text{key} \quad \left. \vphantom{\begin{array}{l} \text{key} = A[j] \\ i = j - 1 \end{array}} \right\} = O(n) \\ \quad \quad A[i+1] = A[i] \\ \quad \quad i = i - 1 \\ \quad A[i+1] = \text{key} \qquad \qquad \qquad = O(1) \end{array} \right.$$

$$\begin{aligned} T(n) &= O(g(n) \times \max(1, 1, n, 1)) \\ &= O(n \times n) \\ &= O(n^2) \end{aligned}$$

Tempo di esecuzione di algoritmi ricorsivi

- **E per gli algoritmi ricorsivi?**
 - Il tempo di esecuzione è espresso tramite una equazione di ricorrenza.

Esempio:

$$\text{Merge Sort: } T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 2T(n/2) + \Theta(n) & \text{se } n > 1 \end{cases}$$

- Sono necessarie tecniche specifiche per risolvere le equazioni di ricorrenza

Algoritmi e Strutture Dati

Tempo di esecuzione di algoritmi ricorsivi

Tempo di esecuzione per algoritmi ricorsivi

Esempio: Fattoriale

```
fact(int n)
  if n <= 1 then
    return 1          /* Caso Base
  else
    return n*fact(n-1) /* Passo Induttivo
```

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

Soluzione di equazioni di ricorrenza

- ***Esistono molto metodi. Ne mostreremo tre:***

Il Metodo Iterativo

- Si itera la regola induttiva di $T(n)$ in termini di n e del caso base.
- Richiede manipolazione delle somme

Il Metodo di Sostituzione

- Si ipotizza una possibile soluzione
- Si sostituisce l'ipotetica soluzione nei casi base e induttivo
- Si dimostra la correttezza della ipotesi tramite induzione matematica

r ***Il Metodo Principale***

Il Metodo Iterativo

Base: $T(1) = a$

Induzione: $T(m) = b + T(m-1)$

I. Sostituire m per $n, n-1, n-2 \dots$ finché si ottiene il caso base

- | | | |
|----------|-----------------------|--------------------------|
| 1) | $T(n) = b + T(n-1)$ | sostituire m con n |
| 2) | $T(n-1) = b + T(n-2)$ | sostituire m con $n-1$ |
| 3) | $T(n-2) = b + T(n-3)$ | sostituire m con $n-2$ |
| | | |
| $n-1$). | $T(2) = b + T(1)$ | sostituire m con 2 |
| | $T(1) = a$ | noto |

Il Metodo Iterativo

II. Sostituire $T(n-1)$, $T(n-2)$... fino al caso base e sostituirlo.

$$\begin{aligned}T(n) &= b + T(n-1) &= \\ & b + b + T(n-2) &= 2*b + T(n-2) &= \\ & 2*b + b + T(n-3) &= 3*b + T(n-3) &= \\ & 3*b + b + T(n-4) &= 4*b + T(n-4) &= \\ & \dots\dots & & \\ & (n-1) * b + T(1) & & \end{aligned}$$

$$T(n) = (n-1) * b + T(1)$$

Inserire il caso base

$$T(n) = (n-1) * b + a$$

III. Valutare l'espressione O-grande associata

$$T(n) = b*n - b + a = O(n)$$

Il Metodo iterativo: Fattoriale

Esempio: Fattoriale

```
fact(int n)
  if n <= 1 then
    return 1          /* Caso Base
  else
    return n*fact(n-1) /* Passo Induttivo
```

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

Equazione di ricorrenza

Base: $T(1) = a$
Induzione: $T(m) = b + T(m-1)$

Il Metodo iterativo: Fattoriale

Analisi di fact

Caso Base: $T(0) = O(1),$
 $T(1) = O(1)$

Passo Induttivo: $O(1) + \max(O(1), T(n-1))$
 $O(1) + T(n-1), \text{ per } n > 1$

Per il fattoriale, l'analisi risulta $T(n) = O(n)$

Il Metodo iterativo: esempio

$$*T(n) = 3 T(n/4) + n*$$

Il Metodo iterativo: esempio

$$\begin{aligned} T(n) &= 3 T(n/4) + n = \\ &= 3 (3 T(n/16) + n/4) + n \end{aligned}$$

Il Metodo iterativo: esempio

$$\begin{aligned} T(n) &= 3 T(n/4) + n = \\ &= 3 (3 T(n/16) + n/4) + n = \\ &= 9 T(n/16) + 3 n/4 + n \end{aligned}$$

Il Metodo iterativo: esempio

$$\begin{aligned} T(n) &= 3 T(n/4) + n = \\ &= 3 (3 T(n/16) + n/4) + n = \\ &= 9 T(n/16) + 3 n/4 + n = \\ &= 27 T(n/64) + 9 n/16 + 3 n/4 + n \end{aligned}$$

Il Metodo iterativo: esempio

$$\begin{aligned}T(n) &= 3 T(n/4) + n = \\ &= 3 (3 T(n/16) + n/4) + n = \\ &= 9 T(n/16) + 3 n/4 + n = \\ &= 27 T(n/64) + 9 n/16 + 3 n/4 + n =\end{aligned}$$

....

Quando ci si ferma?

Il Metodo iterativo: esempio

$$\begin{aligned}T(n) &= 3 T(n/4) + n = \\ &= 3 (3 T(n/16) + n/4) + n = \\ &= 9 T(n/16) + 3 n/4 + n = \\ &= 27 T(n/64) + 9 n/16 + 3 n/4 + n =\end{aligned}$$

.....

Quando ci si ferma?
quando $(n/4)^i = 1$
cioè quando $i > \log_4 n$

$$T(n) < n + 3 n/4 + 9 n/16 + 27 T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

Il Metodo iterativo: esempio

$$T(n) < n + 3n/4 + 9n/16 + 27T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

Contiene una serie geometrica, che è del tipo

$$\sum_{i=0}^n x^i = 1 + x + x^2 + \dots + x^n$$

$$T(n) < n + 3n/4 + 9n/16 + 27T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

$$\leq n \sum_{i=0}^{\infty} (3/4)^i + \Theta(n^{\log_4 3})$$

$$3^{\log_4 n} = n^{\log_4 3}$$

Il Metodo iterativo: esempio

$$T(n) < n + 3n/4 + 9n/16 + 27T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

Contiene una serie geometrica, che è del tipo

$$\sum_{i=0}^n x^i = 1 + x + x^2 + \dots + x^n$$

quando $|x| < 1$ converge a $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$

$$\sum_{i=0}^{\infty} (3/4)^i = \frac{1}{1-3/4} = 4$$

$$T(n) < n + 3n/4 + 9n/16 + 27T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

$$\leq n \sum_{i=0}^{\infty} (3/4)^i + \Theta(n^{\log_4 3})$$

$$3^{\log_4 n} = n^{\log_4 3}$$

Il Metodo iterativo: esempio

$$T(n) < n + 3n/4 + 9n/16 + 27T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

Contiene una serie geometrica, che è del tipo

$$\sum_{i=0}^n x^i = 1 + x + x^2 + \dots + x^n$$

quando $|x| < 1$ converge a $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$

$$\sum_{i=0}^{\infty} (3/4)^i = \frac{1}{1-3/4} = 4$$

$$T(n) < n + 3n/4 + 9n/16 + 27T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

$$\leq n \sum_{i=0}^{\infty} (3/4)^i + \Theta(n^{\log_4 3}) = 4n + o(n)$$

$$3^{\log_4 n} = n^{\log_4 3} \text{ e } \log_4 3 < 1$$

Il Metodo iterativo: esempio

$$T(n) < n + 3n/4 + 9n/16 + 27T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

Contiene una serie geometrica, che è del tipo

$$\sum_{i=0}^n x^i = 1 + x + x^2 + \dots + x^n$$

quando $|x| < 1$ converge a $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$

$$\sum_{i=0}^{\infty} (3/4)^i = \frac{1}{1-3/4} = 4$$

$$T(n) < n + 3n/4 + 9n/16 + 27T(n/64) + \dots + 3^{\log_4 n} \Theta(1)$$

$$\leq n \sum_{i=0}^{\infty} (3/4)^i + \Theta(n^{\log_4 3}) = 4n + o(n)$$

$$= O(n)$$

$$3^{\log_4 n} = n^{\log_4 3} \text{ e } \log_4 3 < 1$$

Metodo iterativo: conclusioni

Importante focalizzarsi su due parametri

- **il numero di volte in cui la ricorrenza deve essere iterata prima di giungere alla condizione limite (o base)**
- **la somma dei termini che compaiono ad ogni livello della iterazione.**

Il Metodo di Sostituzione

Ipotizzare la forma della soluzione, poi usare l'Induzione Matematica per trovare le costanti e dimostrare che l'ipotesi è corretta!

Il Metodo di Sostituzione

Ipotizzare la forma della soluzione, poi usare l'Induzione Matematica per trovare le costanti e dimostrare che l'ipotesi è corretta!

Esempio:

$$T(1) = a$$

$$T(n) = 2T(n/2) + n, \quad \text{per } n > 1$$

Ipotesi:

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

Dim. Per Induzione:

$$T(n) \leq c n \log n,$$

per una scelta opportuna di $c > 0$.

Il Metodo di Sostituzione: MergeSort

```
MergeSort (array A[1...n], Int p, Int r)
```

```
(1)  if p < r then  
(2)    q =  $\lfloor (p+r)/2 \rfloor$   
(3)    MergeSort (A, p, q)  
(4)    MergeSort (A, q+1, r)  
(5)    Merge (A, p, q, r)
```

Il Metodo di Sostituzione: MergeSort

Base: $T(1) = O(1)$

Induzione:

- | | |
|--|-----------|
| 1. $O(1)$ per il test | (linea 1) |
| 2. $O(1)$ (assegnamento) + $O(1)$ (Dividi) | (linea 2) |
| 3. $T(n/2)$ (chiamata MergeSort) | (linea 3) |
| 4. $T(n/2)$ (chiamata MergeSort) | (linea 4) |
| 5. $O(n)$ (chiamata Merge) | (linea 5) |

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

Il Metodo di Sostituzione: MergeSort

Equazione di ricorrenza:

Sostituisci nella notazione O-grande le costanti nascoste

Base: $T(1) = a$

*Induzione: $T(n) = 2 * T(n/2) + b * n$ per n una potenza di 2
e $n > 1$*

Ipotizzare una Soluzione e poi Dimostrare per Induzione.

Il Metodo di Sostituzione: MergeSort

Ipotesi di soluzione:

Considera $T(n)$ per qualche n piccolo (iterativamente):

$$T(2) = 2T(1) + 2b = 2a + 2b$$

$$T(4) = 2T(2) + 4b = 2(2a + 2b) + 4b = 4a + 8b$$

$$T(8) = 2T(4) + 8b = 2(4a + 8b) + 8b = 8a + 24b$$

$$T(16) = 2T(8) + 16b = 2(8a + 24b) + 16b = 16a + 64b$$

$$T(n) = n*a + ?*b$$

Il Metodo di Sostituzione: MergeSort

Ipotesi di soluzione:

Considera $T(n)$ per qualche n piccolo (iterativamente):

$$T(2) = 2T(1) + 2b = \quad \quad \quad = 2a + 2b$$

$$T(4) = 2T(2) + 4b = 2(2a + 2b) + 4b = 4a + 8b$$

$$T(8) = 2T(4) + 8b = 2(4a + 8b) + 8b = 8a + 24b$$

$$T(16) = 2T(8) + 16b = 2(8a + 24b) + 16b = 16a + 64b$$

$$T(n) = n*a + ?*b$$

n	2	4	8	16	
coefficiente di b	2	8	24	64	
iterazioni	1	2	3	4	← $\log n$

Il Metodo di Sostituzione: MergeSort

Ipotesi di soluzione:

Considera $T(n)$ per qualche n piccolo (iterativamente):

$$T(2) = 2T(1) + 2b = 2a + 2b$$

$$T(4) = 2T(2) + 4b = 2(2a + 2b) + 4b = 4a + 8b$$

$$T(8) = 2T(4) + 8b = 2(4a + 8b) + 8b = 8a + 24b$$

$$T(16) = 2T(8) + 16b = 2(8a + 24b) + 16b = 16a + 64b$$

$$T(n) = n*a + ?*b$$

n	2	4	8	16	
coefficiente di b	2	8	24	64	
iterazioni	1	2	3	4	← $\log n$

Ipotesi: $T(n) = a*n + b*n*\log n = O(n \log n)$

Il Metodo di Sostituzione: Prova per Induzione

Dimostrazione:

I. Passo Induttivo

Assumi: il limite vale per $n/2$, cioè
 $T(n/2) \leq c \cdot (n/2) \log(n/2)$

Prova: il limite vale per n :

$$\begin{aligned} T(n) &= 2 T(n/2) + b \cdot n \\ &\leq 2 c \cdot (n/2) \log(n/2) + b \cdot n \end{aligned}$$

Il Metodo di Sostituzione: Prova per Induzione

Dimostrazione:

I. Passo Induttivo

Assumi: il limite vale per $n/2$, cioè
 $T(n/2) \leq c \cdot (n/2) \log(n/2)$

Prova: il limite vale per n :

$$\begin{aligned} T(n) &= 2T(n/2) + b \cdot n \\ &\leq 2c \cdot (n/2) \log(n/2) + b \cdot n \\ &= c \cdot n \log(n/2) + b \cdot n \\ &= c \cdot n \log n - c \cdot n \log 2 + b \cdot n \\ &= c \cdot n \log n - c \cdot n + b \cdot n = c \cdot n \log n - n(c-b) \\ &\leq c \cdot n \log n \quad \text{posto che } c \geq b. \end{aligned}$$

Il Metodo di Sostituzione: Prova per Induzione

II. Passo Base

Bisogna dimostrare :

$$T(1) = a \leq c \cdot 1 \log 1 = 0 \quad \text{Impossibile!}$$

Però, ***O-grande*** richiede che la condizione valga “**per tutti gli $n \geq n_0$** ”, per un qualche **$n_0 > 0$** .

Quindi possiamo scegliere un **n_0 appropriato** per la condizione base (o limite).

In questo caso, possiamo scegliere **$n_0 = 2$** , infatti:

$$T(2) = 2(a + b) < c \cdot 2 \log 2 = 2 \cdot c \quad \text{che vale per } c \geq (a+b)$$

Metodo di Sostituzione: ipotesi di soluzioni

- Tentare soluzioni di casi simili note.

$$T(n) = 2T\left(\frac{n}{2} + 17\right) + n$$

simile a

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

ammettono stessa soluzione $T(n) = O(n \log n)$

- Tentare prima limiti superiori ed inferiori deboli e rafforzarli mano a mano per ridurre l'incertezza

Metodo di Sostituzione: cambio di variabile

$$***T(n) = 2 T(\sqrt{n}) + \log n***$$

Sostituiamo $m = \log n$ (e quindi $n=2^m$) ottenendo

$$***T(2^m) = 2 T(2^{m/2}) + m***$$

Metodo di Sostituzione: cambio di variabile

$$***T(n) = 2 T(\sqrt{n}) + \log n***$$

Sostituiamo $m = \log n$ (e quindi $n=2^m$) ottenendo

$$***T(2^m) = 2 T(2^{m/2}) + m***$$

Poniamo ora $S(m) = T(2^m)$ ottenendo

$$***S(m) = 2 S(m/2) + m***$$

la cui soluzione è $S(m) = O(m \log m)$

Metodo di Sostituzione: cambio di variabile

$$T(n) = 2 T(\sqrt{n}) + \log n$$

Sostituiamo $m = \log n$ (e quindi $n=2^m$) ottenendo

$$T(2^m) = 2 T(2^{m/2}) + m$$

Poniamo ora $S(m) = T(2^m)$ ottenendo

$$S(m) = 2 S(m/2) + m$$

la cui soluzione è $S(m) = O(m \log m)$

Risostituendo all'indietro, otteniamo la soluzione

$$T(n) = T(2^m) = S(m) = O(m \log m) = O(\log n \log \log n)$$