# Analisi

## Cicli for

Si consideri il seguente pezzo di codice:

**for** $i \leftarrow 1$ **to** $n$ **do**
  invoke($i$)

dove l'invocazione invoke($i$) ha costo $O(i^h)$.
Si dimostri, per induzione su $h$, che $\sum_{i=1}^{n} i^h$ è $O(n^{h+1})$.

# Ricorrenza $T(n) = T(n-1) + \log n$

Trovare un limite asintotico superiore e un limite asintotico inferiore alla seguente ricorrenza, facendo uso del metodo di sostituzione:

$$T(n) = \begin{cases} 1 & n \leq 1 \\ T(n-1) + \log n & n > 1 \end{cases}$$

# Ricorrenza

Trovare i limiti superiore e inferiori più stretti possibili per la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 2T(\lfloor n/2 \rfloor) + 4T(\lfloor n/4 \rfloor) + 15T(\lfloor n/8 \rfloor) + n^2 & n > 8 \\ 1 & n \leq 8 \end{cases}$$

# Ricorrenza $4T(\sqrt{n}) + \log^2 n$

Si ottengano limiti superiori e inferiori per la seguente ricorrenza:

$$T(n) = \begin{cases} 4T(\lfloor\sqrt{n}\rfloor) + \log^2 n & n > 1 \\ 1 & n = 1 \end{cases}$$

# Analisi

## SortinoSort

Il professor Sortino ha inventato un nuovo algoritmo di ordinamento. Il vettore di input viene diviso in tre parti, di dimensioni approssimativamente uguali $n/3$. Dopo di che, vengono ordinati ricorsivamente le prime due parti del vettore (ovvero i primi due terzi), i secondi due terzi, e di nuovo i primi due terzi.

---

SortinoSort(**int**[ ] $A$, **int** $i$, **int** $j$)

---

```
if j − i + 1 ≤ 6 then
    InsertionSort(A, i, j)
else
    int s ← ⌈(j − i + 1)/3⌉
    SortinoSort(A, i, i + 2s − 1)
    SortinoSort(A, i + s, j)
    SortinoSort(A, i, i + 2s − 1)
```

---

# Analisi

1. Qual è la complessità di questo algoritmo? Il Prof. Sortino finirà sulla prossima edizione del mio libro?
2. (Difficile, Opzionale) Dimostrare per induzione che questo algoritmo è corretto. Per comodità, assumete pure che tutti i valori siano distinti.

# Spoiler alert!

# Analisi – Cicli For

- Caso base ($h = 0$):

$$\sum_{i=1}^{n} i^0 = \sum_{i=1}^{n} 1 = n = n^{h+1}$$

- Passo induttivo. Supponiamo che la proprietà sia vera per ogni $k < h$; vogliamo dimostrare che la proprietà è vera per $h$:

$$\sum_{i=1}^{n} i^h = \sum_{i=1}^{n} i^{h-1} i \leq \sum_{i=1}^{n} i^{h-1} n = n \sum_{i=1}^{n} i^{h-1} = nO(n^h) = O(n^{h+1})$$

# Ricorrenza $T(n-1)+\log n$

$$T(n) = \begin{cases} 1 & n \leq 1 \\ T(n-1) + \log n & n > 1 \end{cases}$$

Proviamo a dimostrare che $T(n) = O(n \log n)$ per sostituzione.

**Ipotesi Induttiva**: $T(k) \leq ck \log k$ per ogni $k < n$
**Passo induttivo**: vogliamo dimostare che la proprietà è vera per $n$

$$\begin{aligned} T(n) &= T(n-1) + \log n \\ &\leq c(n-1)\log(n-1) + \log n \\ &\leq c(n-1)\log n + \log n \\ &= cn\log n - c\log n + \log n \end{aligned}$$

Abbiamo che $cn \log n - c \log n + \log n \leq cn \log n$ se $c \geq 1$.

# Ricorrenza $T(n-1)+\log n$

$$T(n) = \begin{cases} 1 & n \leq 1 \\ T(n-1) + \log n & n > 1 \end{cases}$$

Proviamo a dimostrare che $T(n) = O(n \log n)$ con il metodo di sostituzione.

**Caso base**:
Il valore 1 non può essere utilizzato perché $\log 1 = 0$.
Ma se prendiamo $T(2) = T(1) + \log 2 = 1 + 1 = 2 \leq c2 \log 2 = 2c$, abbiamo ancora una volta che $c \geq 1$.

# Ricorrenza

E' facile vedere che la ricorrenza è $\Omega(n^2)$, per via della sua componente non ricorsiva. Proviamo quindi a dimostrare che $T(n) = O(n^2)$.

- Caso base: $T(n) = 1 \leq cn^2$, per tutti i valori di $n$ compresi fra 1 e 8. Tutte queste disequazioni sono soddisfatte da $c \geq 1$.
- Ipotesi induttiva: $T(k) \leq ck^2$, per $k < n$
- Passo induttivo:

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + 4T(\lfloor n/4 \rfloor) + 15T(\lfloor n/8 \rfloor) + n^2 \\ &\leq 2c\lfloor n/2 \rfloor^2 + 4c\lfloor n/4 \rfloor^2 + 15\lfloor n/8 \rfloor^2 + n^2 \\ &\leq 2cn^2/4 + 4cn^2/16 + 15cn^2/64 + n^2 \\ &\leq 63/64cn^2 + n^2 \leq cn^2 \end{aligned}$$

L'ultima disequazione è rispettata per $c \geq 64$.

Abbiamo quindi dimostrato che $T(n) = \Theta(n^2)$.

# Ricorrenza $4T(\lfloor\sqrt{n}\rfloor) + \log^2 n$

Poniamo $n = 2^k$. Sostituendo nella ricorrenza otteniamo:

$$\begin{aligned} T(2^k) &= 4T(\sqrt{2^k}) + \log^2 2^k \\ &= 4T(2^{k/2}) + k^2 \end{aligned}$$

Sostituiamo quindi la variable $T(2^k)$ con $S(k)$ e otteniamo (tramite Master Theorem)

$$S(k) = 4S(k/2) + k^2 = \Theta(k^2 \log k)$$

Ri-esprimendo la funzione nei termini di $T(n)$ e $k = \log n$, otteniamo

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

# SortinoSort – 12/09/10

La complessità è rappresentata dalla seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 3T(\frac{2}{3}n) + 1 & n > 6 \\ 1 & n \leq 6 \end{cases}$$

Utilizzando il teorema delle ricorrenze lineari con partizione bilanciata, si ottiene che $a = 3$, $b = 3/2$, da cui $\alpha = \log_{3/2} 3$; inoltre, $\beta = 0$. Siamo quindi nel caso $T(n) = n^{\alpha}$.
Non avete una calcolatrice e non sapete quanto sia $\log_{3/2} 3$?

# SortinoSort – 12/09/10

La complessità è rappresentata dalla seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 3T(\frac{2}{3}n) + 1 & n > 6 \\ 1 & n \leq 6 \end{cases}$$

Utilizzando il teorema delle ricorrenze lineari con partizione bilanciata, si ottiene che $a = 3$, $b = 3/2$, da cui $\alpha = \log_{3/2} 3$; inoltre, $\beta = 0$. Siamo quindi nel caso $T(n) = n^{\alpha}$.
Non avete una calcolatrice e non sapete quanto sia $\log_{3/2} 3$?

E' semplice: $\left(\frac{3}{2}\right)^2 = \frac{9}{4} < 3$, mentre $\left(\frac{3}{2}\right)^3 = \frac{27}{8} > 3$. Quindi $\alpha$ è compreso fra 2 e 3, e quindi questo algoritmo è addirittura peggiore di Insertion Sort. Il prof. Sortino non finirà nel mio libro.