

# Algoritmi e Strutture Dati

Analisi di algoritmi  
Funzioni di costo, notazione asintotica

Alberto Montresor

Università di Trento

2020/09/23

This work is licensed under a Creative Commons  
Attribution-ShareAlike 4.0 International License.



# Notazioni $O$ , $\Omega$ , $\Theta$

## Definizione – Notazione $O$

Sia  $g(n)$  una funzione di costo; indichiamo con  $O(g(n))$  l'insieme delle funzioni  $f(n)$  tali per cui:

$$\exists c > 0, \exists m \geq 0 : f(n) \leq cg(n), \forall n \geq m$$

- Come si legge:  $f(n)$  è “**O grande**” (big-O) di  $g(n)$
- Come si scrive:  $f(n) = O(g(n))$
- $g(n)$  è un **limite asintotico superiore** per  $f(n)$
- $f(n)$  cresce al più come  $g(n)$

# Notazioni $O$ , $\Omega$ , $\Theta$

## Definizione – Notazione $\Omega$

Sia  $g(n)$  una funzione di costo; indichiamo con  $\Omega(g(n))$  l'insieme delle funzioni  $f(n)$  tali per cui:

$$\exists c > 0, \exists m \geq 0 : f(n) \geq cg(n), \forall n \geq m$$

- Come si legge:  $f(n)$  è “**Omega grande**” di  $g(n)$
- Come si scrive:  $f(n) = \Omega(g(n))$
- $g(n)$  è un **limite asintotico inferiore** per  $f(n)$
- $f(n)$  cresce almeno quanto  $g(n)$

# Notazioni $O$ , $\Omega$ , $\Theta$

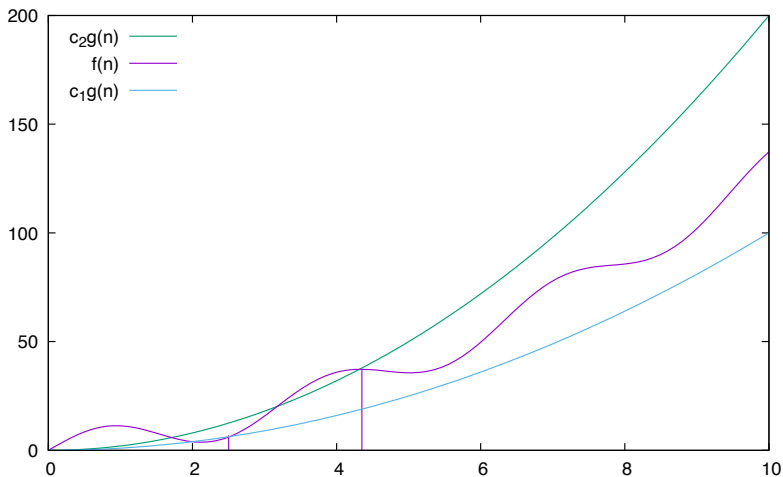
## Definizione – Notazione $\Theta$

Sia  $g(n)$  una funzione di costo; indichiamo con  $\Theta(g(n))$  l'insieme delle funzioni  $f(n)$  tali per cui:

$$\exists c_1 > 0, \exists c_2 > 0, \exists m \geq 0 : c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq m$$

- Come si legge:  $f(n)$  è “**Theta**” di  $g(n)$
- Come si scrive:  $f(n) = \Theta(g(n))$
- $f(n)$  cresce esattamente come  $g(n)$
- $f(n) = \Theta(g(n))$  se e solo se  $f(n) = O(g(n))$  e  $f(n) = \Omega(g(n))$

# Graficamente



# Algoritmi e Strutture Dati

Analisi di algoritmi  
Proprietà della notazione asintotica

Alberto Montresor

Università di Trento

2020/09/23

This work is licensed under a Creative Commons  
Attribution-ShareAlike 4.0 International License.



# Sommario

- 1 Notazione asintotica
  - Definizioni
- 2 Proprietà della notazione asintotica
  - Funzioni di costo particolari
  - Proprietà delle notazioni
  - Altre funzioni di costo
  - Classificazione delle funzioni
- 3 Ricorrenze
  - Introduzione
  - Albero di ricorsione, o per livelli
  - Metodo della sostituzione
  - Metodo dell'esperto
- 4 Back to algorithms!
  - Ruolo dei fattori moltiplicativi

# Regola generale

## Espressioni polinomiali

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0, a_k > 0 \Rightarrow f(n) = \Theta(n^k)$$

**Limite superiore:**  $\exists c > 0, \exists m \geq 0 : f(n) \leq cn^k, \forall n \geq m$

$$\begin{aligned} f(n) &= a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0 \\ &\leq a_k n^k + |a_{k-1}| n^{k-1} + \dots + |a_1| n + |a_0| \\ &\leq a_k n^k + |a_{k-1}| n^k + \dots + |a_1| n^k + |a_0| n^k & \forall n \geq 1 \\ &= (a_k + |a_{k-1}| + \dots + |a_1| + |a_0|) n^k \\ &\stackrel{?}{\leq} cn^k \end{aligned}$$

che è vera per  $c \geq (a_k + |a_{k-1}| + \dots + |a_1| + |a_0|) > 0$  e per  $m = 1$ .



# Regola generale

## Espressioni polinomiali

$$f(n) = a_k n^k + a_{k-1} n^{k-1} + \dots a_1 n + a_0, a_k > 0 \Rightarrow f(n) = \Theta(n^k)$$

**Limite inferiore:**  $\exists d > 0, \exists m \geq 0 : f(n) \geq d n^k, \forall n \geq m$

$$\begin{aligned} f(n) &= a_k n^k + a_{k-1} n^{k-1} + \dots + a_1 n + a_0 \\ &\geq a_k n^k - |a_{k-1}| n^{k-1} - \dots - |a_1| n - |a_0| \\ &\geq a_k n^k - |a_{k-1}| n^{k-1} - \dots - |a_1| n^{k-1} - |a_0| n^{k-1} \quad \forall n \geq 1 \\ &\stackrel{?}{\geq} d n^k \end{aligned}$$

L'ultima equazione è vera se:

$$d \leq a_k - \frac{|a_{k-1}|}{n} - \frac{|a_{k-2}|}{n} - \dots - \frac{|a_1|}{n} - \frac{|a_0|}{n} > 0 \Leftrightarrow n > \frac{|a_{k-1}| + \dots + |a_0|}{a_k}$$

## Alcuni casi particolari

- Qual è la complessità di  $f(n) = 5$ ?

## Alcuni casi particolari

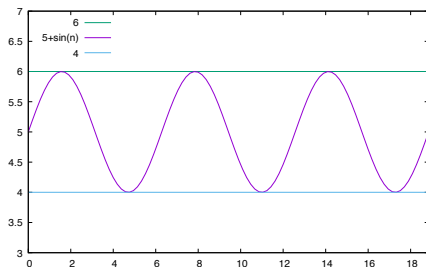
- Qual è la complessità di  $f(n) = 5$ ?
  - $f(n) = 5 \geq c_1 n^0 \Rightarrow c_1 \leq 5$
  - $f(n) = 5 \leq c_2 n^0 \Rightarrow c_2 \geq 5$
  - $f(n) = \Theta(n^0) = \Theta(1)$

## Alcuni casi particolari

- Qual è la complessità di  $f(n) = 5$ ?
  - $f(n) = 5 \geq c_1 n^0 \Rightarrow c_1 \leq 5$
  - $f(n) = 5 \leq c_2 n^0 \Rightarrow c_2 \geq 5$
  - $f(n) = \Theta(n^0) = \Theta(1)$
- Qual è la complessità di  $f(n) = 5 + \sin(n)$ ?

# Alcuni casi particolari

- Qual è la complessità di  $f(n) = 5$ ?
  - $f(n) = 5 \geq c_1 n^0 \Rightarrow c_1 \leq 5$
  - $f(n) = 5 \leq c_2 n^0 \Rightarrow c_2 \geq 5$
  - $f(n) = \Theta(n^0) = \Theta(1)$
- Qual è la complessità di  $f(n) = 5 + \sin(n)$ ?  $\Theta(1)$



# Proprietà

## Dualità

$$f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$$

Dimostrazione:

$$f(n) = O(g(n)) \Leftrightarrow f(n) \leq cg(n), \forall n \geq m$$

$$\Leftrightarrow g(n) \geq \frac{1}{c} f(n), \forall n \geq m$$

$$\Leftrightarrow g(n) \geq c' f(n), \forall n \geq m, c' = \frac{1}{c}$$

$$\Leftrightarrow g(n) = \Omega(f(n))$$

# Proprietà

## Eliminazione delle costanti

$$f(n) = O(g(n)) \Leftrightarrow af(n) = O(g(n)), \forall a > 0$$

$$f(n) = \Omega(g(n)) \Leftrightarrow af(n) = \Omega(g(n)), \forall a > 0$$

Dimostrazione:

$$f(n) = O(g(n)) \Leftrightarrow f(n) \leq \textcolor{red}{c}g(n), \forall n \geq m$$

$$\Leftrightarrow \textcolor{red}{a}f(n) \leq \textcolor{red}{a}c g(n), \forall n \geq m, \forall a \geq 0$$

$$\Leftrightarrow af(n) \leq \textcolor{red}{c'}g(n), \forall n \geq m, \textcolor{red}{c'} = \textcolor{red}{a}c > 0$$

$$\Leftrightarrow af(n) = O(g(n))$$

# Proprietà

## Sommatoria (sequenza di algoritmi)

$$f_1(n) = O(g_1(n)), f_2(n) = O(g_2(n)) \Rightarrow f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$$

$$f_1(n) = \Omega(g_1(n)), f_2(n) = \Omega(g_2(n)) \Rightarrow f_1(n) + f_2(n) = \Omega(\max(g_1(n), g_2(n)))$$

## Dimostrazione (Lato $O$ )

$$f_1(n) = O(g_1(n)) \wedge f_2(n) = O(g_2(n)) \Rightarrow$$

$$f_1(n) \leq c_1 g_1(n) \wedge f_2(n) \leq c_2 g_2(n) \Rightarrow$$

$$f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n) \Rightarrow$$

$$f_1(n) + f_2(n) \leq \max\{c_1, c_2\}(2 \cdot \max(g_1(n), g_2(n))) \Rightarrow$$

$$f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$$



# Proprietà

## Prodotto (Cicli annidati)

$$f_1(n) = O(g_1(n)), f_2(n) = O(g_2(n)) \Rightarrow f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$$

$$f_1(n) = \Omega(g_1(n)), f_2(n) = \Omega(g_2(n)) \Rightarrow f_1(n) \cdot f_2(n) = \Omega(g_1(n) \cdot g_2(n))$$

## Dimostrazione

$$f_1(n) = O(g_1(n)) \wedge f_2(n) = O(g_2(n)) \Rightarrow$$

$$f_1(n) \leq c_1 g_1(n) \wedge f_2(n) \leq c_2 g_2(n) \Rightarrow$$

$$f_1(n) \cdot f_2(n) \leq c_1 c_2 g_1(n) g_2(n)$$

# Proprietà

## Simmetria

$$f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$$

## Dimostrazione

Grazie alla proprietà di dualità:

$$f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n)) \Rightarrow g(n) = \Omega(f(n))$$

$$f(n) = \Theta(g(n)) \Rightarrow f(n) = \Omega(g(n)) \Rightarrow g(n) = O(f(n))$$

# Proprietà

## Transitività

$$f(n) = O(g(n)), g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$

## Dimostrazione

$$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow$$

$$f(n) \leq c_1 g(n) \wedge g(n) \leq c_2 h(n) \Rightarrow$$

$$f(n) \leq c_1 c_2 h(n) \Rightarrow$$

$$f(n) = O(h(n))$$

# Logaritmi vs funzioni lineari

## Proprietà dei logaritmi

Vogliamo provare che  $\log n = O(n)$ . Dimostriamo per induzione che

$$\exists c > 0, \exists m \geq 0 : \log n \leq cn, \forall n \geq m$$

- **Caso base** ( $n = 1$ ):

$$\log 1 = 0 \leq cn = c \cdot 1 \Leftrightarrow c \geq 0$$

# Logaritmi vs funzioni lineari

## Proprietà dei logaritmi

Vogliamo provare che  $\log n = O(n)$ . Dimostriamo per induzione che

$$\exists c > 0, \exists m \geq 0 : \log n \leq cn, \forall n \geq m$$

- **Ipotesi induttiva:** sia  $\log k \leq ck, \forall k \leq n$
- **Passo induttivo:** Dimostriamo la proprietà per  $n + 1$

$$\log(n + 1) \leq \log(n + n) = \log 2n \quad \forall n \geq 1$$

$$= \log 2 + \log n \quad \log ab = \log a + \log b$$

$$= 1 + \log n \quad \log_2 2 = 1$$

$$\leq 1 + cn \quad \text{Per induzione}$$

$$\stackrel{?}{\leq} c(n + 1) \quad \text{Obiettivo}$$

$$1 + cn \leq c(n + 1) \Leftrightarrow c \geq 1$$

# Giocando con le espressioni

- È vero che  $\log_a n = \Theta(\log n)$ ?

# Giocando con le espressioni

- È vero che  $\log_a n = \Theta(\log n)$ ?
  - Sì:  $\log_a n = (\log_a 2) \cdot (\log_2 n) = \Theta(\log n)$

# Giocando con le espressioni

- È vero che  $\log_a n = \Theta(\log n)$ ?
  - Sì:  $\log_a n = (\log_a 2) \cdot (\log_2 n) = \Theta(\log n)$
- È vero che  $\log n^a = \Theta(\log n)$ , per  $a > 0$ ?



# Giocando con le espressioni

- È vero che  $\log_a n = \Theta(\log n)$ ?
  - Sì:  $\log_a n = (\log_a 2) \cdot (\log_2 n) = \Theta(\log n)$
- È vero che  $\log n^a = \Theta(\log n)$ , per  $a > 0$ ?
  - Sì:  $\log n^a = a \log n = \Theta(\log n)$

# Giocando con le espressioni

- È vero che  $\log_a n = \Theta(\log n)$ ?
  - Sì:  $\log_a n = (\log_a 2) \cdot (\log_2 n) = \Theta(\log n)$
- È vero che  $\log n^a = \Theta(\log n)$ , per  $a > 0$ ?
  - Sì:  $\log n^a = a \log n = \Theta(\log n)$
- È vero che  $2^{n+1} = \Theta(2^n)$ ?

# Giocando con le espressioni

- È vero che  $\log_a n = \Theta(\log n)$ ?
  - Sì:  $\log_a n = (\log_a 2) \cdot (\log_2 n) = \Theta(\log n)$
- È vero che  $\log n^a = \Theta(\log n)$ , per  $a > 0$ ?
  - Sì:  $\log n^a = a \log n = \Theta(\log n)$
- È vero che  $2^{n+1} = \Theta(2^n)$ ?
  - Sì:  $2^{n+1} = 2 \cdot 2^n = \Theta(2^n)$

# Giocando con le espressioni

- È vero che  $\log_a n = \Theta(\log n)$ ?
  - Sì:  $\log_a n = (\log_a 2) \cdot (\log_2 n) = \Theta(\log n)$
- È vero che  $\log n^a = \Theta(\log n)$ , per  $a > 0$ ?
  - Sì:  $\log n^a = a \log n = \Theta(\log n)$
- È vero che  $2^{n+1} = \Theta(2^n)$ ?
  - Sì:  $2^{n+1} = 2 \cdot 2^n = \Theta(2^n)$
- È vero che  $2^n = \Theta(3^n)$ ?

# Giocando con le espressioni

- È vero che  $\log_a n = \Theta(\log n)$ ?
  - Sì:  $\log_a n = (\log_a 2) \cdot (\log_2 n) = \Theta(\log n)$
- È vero che  $\log n^a = \Theta(\log n)$ , per  $a > 0$ ?
  - Sì:  $\log n^a = a \log n = \Theta(\log n)$
- È vero che  $2^{n+1} = \Theta(2^n)$ ?
  - Sì:  $2^{n+1} = 2 \cdot 2^n = \Theta(2^n)$
- È vero che  $2^n = \Theta(3^n)$ ?
  - Ovviamente  $2^n = O(3^n)$
  - Ma:  $3^n = \left(\frac{3}{2} \cdot 2\right)^n = \left(\frac{3}{2}\right)^n \cdot 2^n$ :  
Quindi non esiste  $c > 0$  tale per cui  $\left(\frac{3}{2}\right)^n \cdot 2^n \leq c2^n$ , e quindi  
 $2^n \neq \Omega(3^n)$

# Notazioni $o, \omega$

## Definizione – Notazioni $o, \omega$

Sia  $g(n)$  una funzione di costo; indichiamo con  $o(g(n))$  l'insieme delle funzioni  $f(n)$  tali per cui:

$$\forall c, \exists m : f(n) < cg(n), \forall n \geq m.$$

Sia  $g(n)$  una funzione di costo; indichiamo con  $\omega(g(n))$  l'insieme delle funzioni  $f(n)$  tali per cui:

$$\forall c, \exists m : f(n) > cg(n), \forall n \geq m.$$

- Come si leggono:  $f(n)$  è “**o piccolo**”, “**omega piccolo**” di  $g(n)$
- Come si scrivono:  $f(n) = o(g(n))$  oppure  $f(n) = \omega(g(n))$

## Notazioni $o, \omega$

Utilizzando il concetto di limite, date due funzioni  $f(n)$  e  $g(n)$  si possono fare le seguenti affermazioni:

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) = o(g(n))$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = c \neq 0 \Rightarrow f(n) = \Theta(g(n))$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = +\infty \Rightarrow f(n) = \omega(g(n))$$

Si noti che:

$$f(n) = o(g(n)) \Rightarrow f(n) = O(g(n))$$

$$f(n) = \omega(g(n)) \Rightarrow f(n) = \Omega(g(n))$$

# Classificazione delle funzioni

E' possibile trarre un'ordinamento delle principali espressioni, estendendo le relazioni che abbiamo dimostrato fino ad ora

Per ogni  $r < s, h < k, a < b$ :

$$O(1) \subset O(\log^r n) \subset O(\log^s n) \subset O(n^h) \subset O(n^h \log^r n) \subset O(n^h \log^s n) \subset O(n^k) \subset O(a^n) \subset O(b^n)$$



# Algoritmi e Strutture Dati

Analisi di algoritmi  
Ricorrenze, metodo dell'albero di ricorsione

Alberto Montresor

Università di Trento

2020/09/23

This work is licensed under a Creative Commons  
Attribution-ShareAlike 4.0 International License.



# Sommario

- 1 Notazione asintotica
  - Definizioni
- 2 Proprietà della notazione asintotica
  - Funzioni di costo particolari
  - Proprietà delle notazioni
  - Altre funzioni di costo
  - Classificazione delle funzioni
- 3 Ricorrenze
  - Introduzione
  - Albero di ricorsione, o per livelli
  - Metodo della sostituzione
  - Metodo dell'esperto
- 4 Back to algorithms!
  - Ruolo dei fattori moltiplicativi

# Introduzione

## Equazioni di ricorrenza

Quando si calcola la complessità di un algoritmo ricorsivo, questa viene espressa tramite un'**equazione di ricorrenza**, ovvero una formula matematica definita in maniera... ricorsiva!

## MergeSort

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) & n > 1 \\ \Theta(1) & n \leq 1 \end{cases}$$

# Introduzione

## Forma chiusa

Il nostro obiettivo è ottenere, quando possibile, una **formula chiusa** che rappresenti la classe di complessità della funzione.

## MergeSort

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) & n > 1 \\ \Theta(1) & n \leq 1 \end{cases}$$

# Introduzione

## Forma chiusa

Il nostro obiettivo è ottenere, quando possibile, una **formula chiusa** che rappresenti la classe di complessità della funzione.

## MergeSort

$$T(n) = \Theta(n \log n)$$

# Oltre l'analisi di algoritmi

Utilizzeremo le equazioni di ricorrenza anche per risolvere problemi

## Problema

Un bambino scende una scala composta da  $n$  scalini. Ad ogni passo, può decidere di fare 1,2,3,4 scalini alla volta. Determinare in quanti modi diversi può scendere le scale. Ad esempio, se  $n = 7$ , alcuni dei modi possibili sono i seguenti:

- 1,1,1,1,1,1,1
- 1,2,4
- 4,2,1
- 2,2,2,1
- 1,2,2,1,1

# Oltre l'analisi di algoritmi

## Soluzione

Sia  $M(n)$  il numero di modi in cui è possibile scegliere  $n$  scalini; allora  $M(n)$  può essere espresso nel modo seguente:

$$M(n) = \begin{cases} 0 & n < 0 \\ 1 & n = 0 \\ \sum_{k=1}^4 M(n-k) & n > 0 \end{cases}$$

Questa ricorrenza può essere trasformata in un algoritmo tramite semplice ricorsione o tramite programmazione dinamica.

## Numeri di Tetranacci

1, 1, 2, 4, 8, 15, 29, 56, 108, 208, 401, 773, 1490, 2872, 5536, ...

# Metodo dell'albero di ricorsione, o per livelli

## Metodi per risolvere ricorrenze

- **Analisi per livelli**
- Analisi per tentativi, o per sostituzione
- Metodo dell'esperto, o delle ricorrenze comuni

## Metodo dell'albero di ricorsione, o per livelli

“Srotoliamo” la ricorrenza in un albero i cui nodi rappresentano i costi ai vari livelli della ricorsione



## Primo esempio

$$T(n) = \begin{cases} T(n/2) + b & n > 1 \\ c & n \leq 1 \end{cases}$$

È possibile risolvere questa ricorrenza nel modo seguente:

$$\begin{aligned} T(n) &= b + T(n/2) \\ &= b + b + T(n/4) \\ &= b + b + b + T(n/8) \\ &= \dots \\ &= \underbrace{b + b + \dots + b}_{\log n} + T(1) \end{aligned}$$

Assumiamo per semplicità:  
 $n = 2^k$ , ovvero  $k = \log n$

## Primo esempio

$$T(n) = \begin{cases} T(n/2) + b & n > 1 \\ c & n \leq 1 \end{cases}$$

È possibile risolvere questa ricorrenza nel modo seguente:

$$\begin{aligned} T(n) &= b + T(n/2) \\ &= b + b + T(n/4) \\ &= b + b + b + T(n/8) \\ &= \dots \\ &= \underbrace{b + b + \dots + b}_{\log n} + T(1) \end{aligned}$$

Assumiamo per semplicità:  
 $n = 2^k$ , ovvero  $k = \log n$

$$T(n) = b \log n + c = \Theta(\log n)$$

## Secondo esempio

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

È possibile risolvere questa ricorrenza nel modo seguente:

$$\begin{aligned} T(n) &= n + 4T(n/2) \\ &= n + 4n/2 + 16T(n/2^2) \\ &= n + 2n + 16n/4 + 64T(n/8) \\ &= \dots \\ &= n + 2n + 4n + 8n + \dots + 2^{\log n - 1}n + 4^{\log n}T(1) \\ &= n \sum_{j=0}^{\log n - 1} 2^j + 4^{\log n} \end{aligned}$$

## Secondo esempio

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

È possibile risolvere questa ricorrenza nel modo seguente:

$$T(n) = n \sum_{j=0}^{\log n - 1} 2^j + 4^{\log n}$$

## Secondo esempio

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

È possibile risolvere questa ricorrenza nel modo seguente:

$$T(n) = n \sum_{j=0}^{\log n - 1} 2^j \quad n \cdot \frac{2^{\log n} - 1}{2 - 1} \quad + 4^{\log n}$$

Serie geometrica finita:

$$\forall x \neq 1 : \sum_{j=0}^k x^j = \frac{x^{k+1} - 1}{x - 1}$$

## Secondo esempio

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

È possibile risolvere questa ricorrenza nel modo seguente:

$$T(n) = n \sum_{j=0}^{\log n - 1} 2^j \quad n \cdot \frac{2^{\log n} - 1}{2 - 1} \quad n(n-1) + 4^{\log n}$$

Passaggi algebrici

## Secondo esempio

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

È possibile risolvere questa ricorrenza nel modo seguente:

$$T(n) = n \sum_{j=0}^{\log n - 1} 2^j \quad \cancel{n \cdot \frac{2^{\log n} - 1}{2 - 1}} \quad n(n-1) + \cancel{4^{\log n}} \quad n^{\log 4}$$

Cambiamento di base:

$$\log_b n = (\log_b a) \cdot (\log_a n) \Rightarrow$$

$$a^{\log_b n} = n^{\log_b a}$$

## Secondo esempio

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

È possibile risolvere questa ricorrenza nel modo seguente:

$$\begin{aligned}
 T(n) &= n \sum_{j=0}^{\log n - 1} 2^j \quad n \cdot \frac{2^{\log n} - 1}{2 - 1} \quad n(n - 1) + \cancel{4^{\log n}} \quad \cancel{n^{\log 4}} \quad n^2 \\
 &= 2n^2 - n = \Theta(n^2)
 \end{aligned}$$



## Terzo esempio

$$T(n) = \begin{cases} 4T(n/2) + n^3 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Proviamo a visualizzare l'albero delle chiamate, per i primi tre livelli:

$$\begin{array}{c}
 \overbrace{\left(\frac{n}{2}\right)^3 \quad \left(\frac{n}{2}\right)^3 \quad \left(\frac{n}{2}\right)^3 \quad \left(\frac{n}{2}\right)^3}^{n^3} \\
 \overbrace{\left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3}^{\left(\frac{n}{2}\right)^3} \quad \overbrace{\left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3}^{\left(\frac{n}{2}\right)^3} \quad \overbrace{\left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3}^{\left(\frac{n}{2}\right)^3} \quad \overbrace{\left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3 \left(\frac{n}{4}\right)^3}^{\left(\frac{n}{2}\right)^3}
 \end{array}$$

## Terzo esempio

$$T(n) = \begin{cases} 4T(n/2) + n^3 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Livello	Dim.	Costo chiam.	N. chiamate	Costo livello
0	$n$	$n^3$	1	$n^3$
1	$n/2$	$(n/2)^3$	4	$4(n/2)^3$
2	$n/4$	$(n/4)^3$	16	$16(n/4)^3$
...	...	...	...	...
$i$	$n/2^i$	$(n/2^i)^3$	$4^i$	$4^i(n/2^i)^3$
...	...	...	...	...
$\ell - 1$	$n/2^{\ell-1}$	$(n/2^{\ell-1})^3$	$4^{\ell-1}$	$4^{\ell-1}(n/2^{\ell-1})^3$
$\ell = \log n$	1	$T(1)$	$4^{\log n}$	$4^{\log n}$

## Terzo esempio

$$T(n) = \begin{cases} 4T(n/2) + n^3 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

La sommatoria dà origine a:

$$T(n) = \sum_{i=0}^{\log n - 1} 4^i \cdot n^3 / 2^{3i} + 4^{\log n}$$

$$= n^3 \sum_{i=0}^{\log n - 1} \frac{2^{2i}}{2^{3i}} + 4^{\log n}$$

$$= n^3 \sum_{i=0}^{\log n - 1} \left(\frac{1}{2}\right)^i + 4^{\log n}$$

Passaggi algebrici

Passaggi algebrici

## Terzo esempio

$$T(n) = \begin{cases} 4T(n/2) + n^3 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

La sommatoria dà origine a:

$$T(n) = n^3 \sum_{i=0}^{\log n - 1} \left(\frac{1}{2}\right)^i + 4^{\log n}$$

$$= n^3 \sum_{i=0}^{\log n - 1} \left(\frac{1}{2}\right)^i + n^2$$

$$\leq n^3 \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i + n^2$$

Cambiamento di base

Estensione della sommatoria

## Terzo esempio

$$T(n) = \begin{cases} 4T(n/2) + n^3 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

La sommatoria dà origine a:

$$\begin{aligned} T(n) &\leq n^3 \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i + n^2 \\ &= n^3 \cdot \frac{1}{1 - \frac{1}{2}} + n^2 \\ &= 2n^3 + n^2 \end{aligned}$$

Serie geometrica infinita decrescente:

$$\forall x, |x| < 1 : \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$$

## Terzo esempio

$$T(n) = \begin{cases} 4T(n/2) + n^3 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Abbiamo dimostrato che:

$$T(n) \leq 2n^3 + n^2$$

- Possiamo affermare che  $T(n) = O(n^3)$
- La dimostrazione precedente non afferma che  $T(n) = \Theta(n^3)$ , perché ad un certo punto siamo passati a  $\leq$
- Però è possibile notare che  $T(n) \geq n^3$ , quindi è possibile affermare che  $T(n) = \Omega(n^3)$  e quindi  $T(n) = \Theta(n^3)$

## Quarto esempio

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

Livello	Dimensione	Costo chiamata	N. chiamate	Costo livello
0	$n$	$n^2$	1	$n^2$
1	$n/2$	$(n/2)^2$	4	$4(n/2)^2$
2	$n/4$	$(n/4)^2$	16	$16(n/4)^2$
...	...	...	...	...
$i$	$n/2^i$	$(n/2^i)^2$	$4^i$	$4^i(n/2^i)^2$
...	...	...	...	...
$\ell - 1$	$n/2^{\ell-1}$	$(n/2^{\ell-1})^2$	$4^{\ell-1}$	$4^{\ell-1}(n/2^{\ell-1})^2$
$\ell = \log n$	1	$T(1)$	$4^{\log n}$	$4^{\log n}$

## Quarto esempio

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

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log n - 1} n^2 / 2^{2i} \cdot 4^i + 4^{\log n} \\ &= n^2 \sum_{i=0}^{\log n - 1} \frac{2^{2i}}{2^{2i}} + n^2 \\ &= n^2 \sum_{i=0}^{\log n - 1} 1 + n^2 \\ &= n^2 \log n + n^2 = \Theta(n^2 \log n) \end{aligned}$$



# Algoritmi e Strutture Dati

Analisi di algoritmi  
Ricorrenze, metodo di sostituzione

Alberto Montresor

Università di Trento

2020/09/23

This work is licensed under a Creative Commons  
Attribution-ShareAlike 4.0 International License.



# Metodo della sostituzione

## Metodi per risolvere ricorrenze

- Metodo dell'albero di ricorsione, o per livelli
- **Metodo di sostituzione, o per tentativi**
- Metodo dell'esperto, o delle ricorrenze comuni

## Metodo di sostituzione

È un metodo in cui si cerca di “**indovinare**” (guess) una soluzione, in base alla propria esperienza, e si dimostra che questa soluzione è corretta tramite **induzione**.

## Cerchiamo di indovinare

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

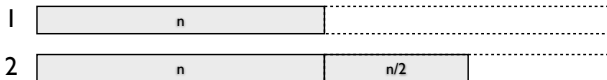
# Cerchiamo di indovinare

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



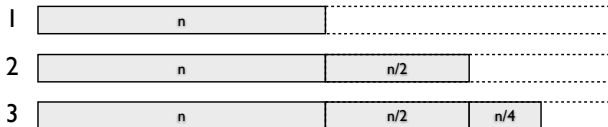
# Cerchiamo di indovinare

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



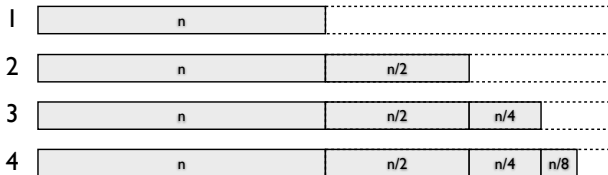
# Cerchiamo di indovinare

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



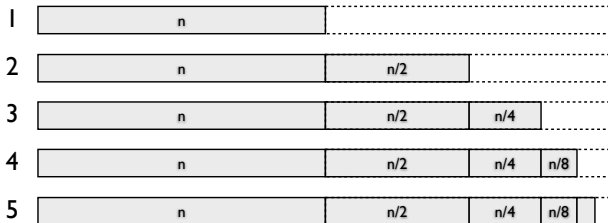
# Cerchiamo di indovinare

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



# Cerchiamo di indovinare

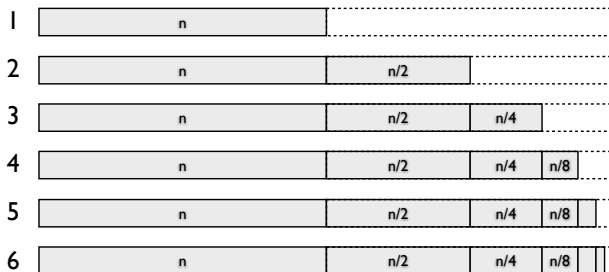
$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases}$$





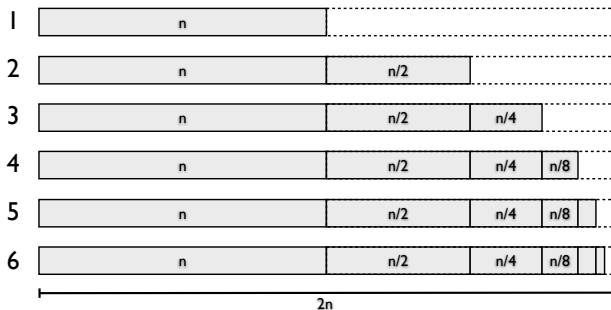
# Cerchiamo di indovinare

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



# Cerchiamo di indovinare

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



## Cerchiamo di indovinare

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

$$T(n) = n \cdot \sum_{i=0}^{\log n} (1/2)^i \leq n \cdot \sum_{i=0}^{\infty} (1/2)^i = n \cdot \frac{1}{1 - \frac{1}{2}} = 2n$$

### Serie geometrica decrescente infinita

$$\forall x, |x| < 1 : \sum_{i=0}^{\infty} x^i = \frac{1}{1 - x}$$

# Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \text{Tentativo: } T(n) = O(n)$$
$$\exists c > 0, \exists m \geq 0 : T(n) \leq cn, \forall n \geq m$$

- **Caso base:** Dimostriamo la disequazione per  $T(1)$

$$T(1) = 1 \stackrel{?}{\leq} 1 \cdot c \Leftrightarrow \forall c \geq 1$$

# Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \begin{array}{l} \text{Tentativo: } T(n) = O(n) \\ \exists c > 0, \exists m \geq 0 : T(n) \leq cn, \forall n \geq m \end{array}$$

- **Ipotesi induttiva:**  $\forall k < n : T(k) \leq ck$ .
- **Passo di induzione:** Dimostriamo la disequazione per  $T(n)$

$$T(n) = T(\lfloor n/2 \rfloor) + n$$

$$\leq c\lfloor n/2 \rfloor + n$$

$$\leq cn/2 + n$$

$$= (c/2 + 1)n$$

?

$$\leq cn$$

$$\Leftrightarrow c/2 + 1 \leq c \Leftrightarrow c \geq 2$$

Sostituzione

Intero inferiore

Passo algebrico

Obiettivo

Risultato finale

# Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \text{Tentativo: } T(n) = O(n) \quad \exists c > 0, \exists m \geq 0 : T(n) \leq cn, \forall n \geq m$$

- Abbiamo provato che  $T(n) \leq cn$ 
  - Nel caso base:  $c \geq 1$
  - Nel passo induttivo:  $c \geq 2$
  - Un valore  $c$  che rispetta entrambe le disequazioni è  $c = 2$
- Questo vale per  $n = 1$ , e per tutti i valori di  $n$  seguenti
  - Quindi  $m = 1$

Abbiamo quindi provato che  $T(n) = O(n)$

# Limite inferiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \begin{array}{l} \text{Tentativo: } T(n) = \Omega(n) \\ \exists d > 0, \exists m \geq 0 : T(n) \geq dn, \forall n \geq m \end{array}$$

- **Caso base:** Dimostriamo la disequazione per  $T(1)$

$$T(1) = 1 \stackrel{?}{\geq} 1 \cdot d \Leftrightarrow \forall d \leq 1$$

# Limite inferiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \begin{array}{l} \text{Tentativo: } T(n) = \Omega(n) \\ \exists d > 0, \exists m \geq 0 : T(n) \geq dn, \forall n \geq m \end{array}$$

- **Ipotesi induttiva:**  $\forall k < n : T(k) \geq dk$ .
- **Passo di induzione:** Dimostriamo la disequazione per  $T(n)$

$$T(n) = T(\lfloor n/2 \rfloor) + n$$

$$\geq d \lfloor n/2 \rfloor + n$$

Sostituzione

$$\geq dn/2 - 1 + n$$

Intero inferiore

$$= \left( \frac{d}{2} - \frac{1}{n} + 1 \right) n \stackrel{?}{\geq} dn$$

Passo algebrico

$$\Leftrightarrow \frac{d}{2} - \frac{1}{n} + 1 \geq d \Leftrightarrow d \leq 2 - 2/n$$

Risultato finale



# Limite inferiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \begin{array}{l} \text{Tentativo: } T(n) = \Omega(n) \\ \exists d > 0, \exists m \geq 0 : T(n) \geq dn, \forall n \geq m \end{array}$$

- Abbiamo provato che  $T(n) \geq dn$ 
  - Nel caso base:  $d \leq 1$
  - Nel passo induttivo:  $d \leq 2 - \frac{2}{n}$
  - Un valore  $d$  che rispetta entrambe le disequazioni, per ogni valore di  $n \geq 2$ , è  $d = 1$
- Questo vale per  $n = 2$ , e per tutti i valori di  $n$  seguenti
  - Quindi  $m = 2$

Abbiamo quindi provato che  $T(n) = \Omega(n)$

$$T(n) = O(n) \wedge T(n) = \Omega(n) \Leftrightarrow T(n) = \Theta(n)$$

# Limite inferiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \begin{array}{l} \text{Tentativo: } T(n) = \Omega(n) \\ \exists d > 0, \exists m \geq 0 : T(n) \geq dn, \forall n \geq m \end{array}$$

- È possibile dimostrare che  $T(n) = \Omega(n)$  in maniera molto più semplice, senza fare nemmeno ricorso all'ipotesi induttiva.

$$T(n) = T(\lfloor n/2 \rfloor) + n \stackrel{?}{\geq} n \geq dn$$

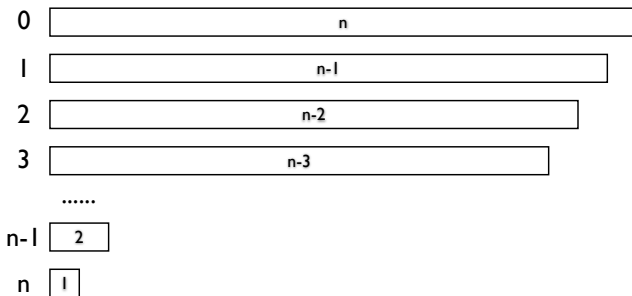
- L'ultima equazione è vera per  $d \leq 1$ , condizione identica a quella del caso base

## Cosa succede se si sbaglia l'intuizione

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

# Cosa succede se si sbaglia l'intuizione

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



## Cosa succede se si sbaglia l'intuizione

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

$$T(n) = \sum_{i=1}^n i = \frac{n(n+1)}{2} = \Theta(n^2)$$

# Cosa succede se si sbaglia l'intuizione

$$T(n) = \begin{cases} T(n-1) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \begin{array}{l} \text{Tentativo sbagliato: } T(n) = O(n) \\ \exists c > 0, \exists m \geq 0 : T(n) \leq cn, \forall n \geq m \end{array}$$

- **Ipotesi induttiva:**  $\forall k < n : T(k) \leq ck$ .
- **Passo di induzione:** Dimostriamo la disequazione per  $T(n)$

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

$$\leq c(n-1) + n$$

$$\leq (c+1)n - c$$

$$\stackrel{?}{\leq} cn$$

$$\Rightarrow n - c \leq 0$$

$$\Rightarrow c \geq n$$

Sostituzione

Passo algebrico

Obiettivo

Semplificando  $cn$

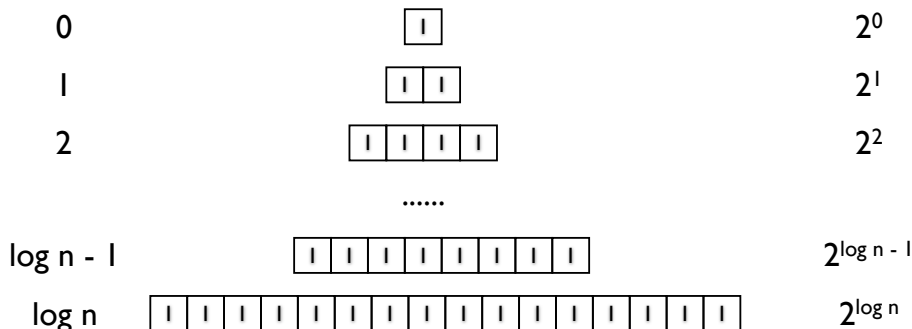
$c$  non è costante

# Difficoltà matematica – Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

# Difficoltà matematica – Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$





# Difficoltà matematica – Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

$$T(n) = \sum_{i=0}^{\log n} 2^i = 1 + 2 + \dots + n/4 + n/2 + n = O(n)$$

# Difficoltà matematica – Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

**Tentativo:**  $T(n) = O(n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn, \forall n \geq m$

- **Ipotesi induttiva:**  $\forall k < n : T(k) \leq ck$ .
- **Passo di induzione:** Dimostriamo la disequazione per  $T(n)$

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

$$\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1$$

$$= cn + 1$$

$$\stackrel{?}{\leq} cn$$

$$\Rightarrow 1 \leq 0$$

Sostituzione

Passo algebrico

Obiettivo

Impossibile

# Difficoltà matematica – Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Tentativo:  $T(n) = O(n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn, \forall n \geq m$

## Cosa succede?

- Il tentativo è corretto...
- ma non riusciamo a dimostrarlo per **un termine di ordine inferiore**

$$cn + 1 \leq cn$$

# Difficoltà matematica – Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

**Tentativo:**  $T(n) = O(n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn, \forall n \geq m$

- **Ipotesi induttiva più stretta:**  $\exists b > 0, \forall k < n : T(k) \leq ck - b$ .
- **Passo di induzione:** Dimostriamo la disequazione per  $T(n)$ :

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

$$\leq c\lfloor n/2 \rfloor - b + c\lceil n/2 \rceil - b + 1$$

$$= cn - 2b + 1$$

$$\stackrel{?}{\leq} cn - b$$

$$\Rightarrow -2b + 1 \leq -b$$

$$\Rightarrow b \geq 1$$

Sostituzione

Passo algebrico

Obiettivo

Eliminazione  $cn$ 

Passo algebrico

# Difficoltà matematica – Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

Tentativo:  $T(n) = O(n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn, \forall n \geq m$

- **Caso base:** Dimostriamo la disequazione per  $T(1)$

$$T(1) = 1 \stackrel{?}{\leq} 1 \cdot c - b \Leftrightarrow \forall c \geq b + 1$$

# Difficoltà matematica – Limite superiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

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

$\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn, \forall n \geq m$

- Abbiamo provato che  $T(n) \leq cn - b \leq cn$ 
  - Nel passo induttivo:  $\forall b \geq 1, \forall c$
  - Nel caso base:  $\forall c \geq b + 1$
  - Una coppia di valori  $b, c$  che rispettano queste disequazioni sono  $b = 1, c = 2$
- Questo vale per  $n = 1$ , e per tutti i valori di  $n$  seguenti
  - Quindi  $m = 1$

Abbiamo quindi provato che  $T(n) = O(n)$

# Difficoltà matematica – Limite inferiore

$$T(n) = \begin{cases} T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 & n > 1 \\ 1 & n \leq 1 \end{cases}$$

**Tentativo:**  $T(n) = \Omega(n)$   
 $\exists d > 0, \exists m \geq 0 :$   
 $T(n) \geq dn, \forall n \geq m$

- **Passo di induzione:** Dimostriamo la disequazione per  $T(n)$

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1 \\ &\geq d\lfloor n/2 \rfloor + d\lceil n/2 \rceil + 1 \\ &= dn + 1 \stackrel{?}{\geq} dn \end{aligned}$$

Sostituzione

Vero per ogni  $d$ 

- **Caso base:** Dimostriamo la disequazione per  $T(1)$

$$T(1) = 1 \geq d \cdot 1 \Leftrightarrow d \leq 1$$

Abbiamo quindi provato che  $T(n) = \Omega(n)$

# Problemi con i casi base

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

0 

n
---

1 

n/2	n/2
-----	-----

2 

n/4	n/4	n/4	n/4
-----	-----	-----	-----

.....

log n - 1 

2	2	2	2	2	2	2	2	2	2
---	---	---	---	---	---	---	---	---	---

log n 

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



## Problemi con i casi base

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

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

# Problemi con i casi base

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

**Tentativo:**  $T(n) = O(n \log n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn \log n, \forall n \geq m$

- **Ipotesi induttiva:**  $\exists c > 0, \forall k < n : T(k) \leq ck \log k$ .
- **Passo di induzione:** Dimostriamo la disequazione per  $T(n)$ :

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &\leq 2c\lfloor n/2 \rfloor \log \lfloor n/2 \rfloor + n \\ &\leq 2cn/2 \log n/2 + n \\ &= cn(\log n - 1) + n \\ &= cn \log n - cn + n \\ &\stackrel{?}{\leq} cn \log n \end{aligned}$$

Sostituzione

Intero inferiore

Passo algebrico

Passo algebrico

Obiettivo

# Problemi con i casi base

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

**Tentativo:**  $T(n) = O(n \log n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn \log n, \forall n \geq m$

- **Ipotesi induttiva:**  $\exists c > 0, \forall k < n : T(k) \leq ck \log k$ .
- **Passo di induzione:** Dimostriamo la disequazione per  $T(n)$ :

$$T(n) \leq cn \log n - cn + n \stackrel{?}{\leq} cn \log n$$

$$\Rightarrow -cn + n \leq 0$$

$$\Rightarrow c \geq 1$$

Obiettivo

Eliminazione  $cn \log n$

Passo algebrico

# Problemi con i casi base

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

**Tentativo:**  $T(n) = O(n \log n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn \log n, \forall n \geq m$

- **Caso base:** Dimostriamo la disequazione per  $T(1)$

$$T(1) = 1 \stackrel{?}{\leq} 1 \cdot c \log 1 = 0 \Rightarrow 1 \not\leq 0$$

# Problemi con i casi base

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

Tentativo:  $T(n) = O(n \log n)$

$\exists c > 0, \exists m \geq 0 :$

$T(n) \leq cn \log n, \forall n \geq m$

## Cosa succede?

- È falso, ma non è un problema: non a caso si chiama notazione asintotica.
- Il valore iniziale di  $m$  lo possiamo scegliere noi

## Problemi con i casi base

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

**Tentativo:**  $T(n) = O(n \log n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn \log n, \forall n \geq m$

- **Caso base:** Dimostriamo la disequazione per  $T(2)$ ,  $T(3)$ :

$$T(2) = 2T(\lfloor 2/2 \rfloor) + 2 = 4 \leq 1 \cdot c \cdot 2 \log 2 \Leftrightarrow c \geq 2$$

$$T(3) = 2T(\lfloor 3/2 \rfloor) + 3 = 5 \leq 1 \cdot c \cdot 3 \log 3 \Leftrightarrow c \geq \frac{5}{3 \log 3}$$

$$T(4) = 2T(\lfloor 4/2 \rfloor) + 4 = 2T(\lfloor 2 \rfloor) + 4$$

- Non è necessario provare la terza disequazione, in quanto viene espressa in base a casi base diversi da  $T(1)$  che sono già stati dimostrati e quindi possono costituire la base della nostra induzione.

# Problemi con i casi base

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

Tentativo:  $T(n) = O(n \log n)$   
 $\exists c > 0, \exists m \geq 0 :$   
 $T(n) \leq cn \log n, \forall n \geq m$

- Abbiamo provato che  $T(n) \leq cn \log n$ 
  - Nel passo induttivo:  $\forall c \geq 1$
  - Nel caso base:  $\forall c \geq 2, c \geq \frac{5}{3 \log 3}$
  - Visto che sono tutte disequazioni con il segno  $\geq$ , è sufficiente utilizzare un valore  $c \geq \max\{1, 2, \frac{5}{3 \log 3}\}$
- Questo vale per  $n = 2, n = 3$ , e per tutti i valori di  $n$  seguenti
  - Quindi  $m = 2$

together now!

$$T(n) = \begin{cases} 9T(\lfloor n/3 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \text{Tentativo: } T(n) = O(n^2)$$

$$\exists c > 0, \exists m \geq 0 : T(n) \leq cn^2, \forall n \geq m$$

- **Ipotesi induttiva:**  $\exists c > 0 : T(k) \leq ck^2, \forall k < n$
- **Passo induttivo:** Dimostriamo la disequazione per  $T(n)$

$$T(n) = 9T(\lfloor n/3 \rfloor) + n$$

$$\leq 9c(\lfloor n/3 \rfloor)^2 + n$$

$$\leq 9c(n^2/9) + n$$

$$= cn^2 + n$$

$$\not\leq cn^2$$

Sostituzione

Limite inferiore

Passo algebrico

Falso



together now!

$$T(n) = \begin{cases} 9T(\lfloor n/3 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \text{Tentativo: } T(n) = O(n^2)$$

$$\exists c > 0, \exists m \geq 0 : T(n) \leq cn^2, \forall n \geq m$$

- **Ipotesi induttiva:**  $\exists c > 0 : T(k) \leq c(k^2 - k), \forall k < n$
- **Passo induttivo:** Dimostriamo la disequazione per  $T(n)$

$$\begin{aligned} T(n) &= 9T(\lfloor n/3 \rfloor) + n \\ &\leq 9c(\lfloor n/3 \rfloor^2 - \lfloor n/3 \rfloor) + n \\ &\leq cn^2 - 3cn + n \\ &\stackrel{?}{\leq} cn^2 - cn \\ &\Leftrightarrow c \geq \frac{1}{2} \end{aligned}$$

Sostituzione

Limite inferiore

Obiettivo

together now!

$$T(n) = \begin{cases} 9T(\lfloor n/3 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \text{Tentativo: } T(n) = O(n^2)$$

$\exists c > 0, \exists m \geq 0 : T(n) \leq cn^2, \forall n \geq m$

- **Ipotesi induttiva:**  $\exists c > 0 : T(k) \leq c(k^2 - k), \forall k < n$
- **Passo base:**
  - $T(1) = 1 \leq c(1^2 - 1) = 0$ , falso
  - $T(2) = 9T(0) + 2 = 11 \leq c(4 - 2) \Leftrightarrow c \geq 11/2$
  - $T(3) = 9T(1) + 3 = 12 \leq c(9 - 3) \Leftrightarrow c \geq 12/6$
  - $T(4) = 9T(1) + 4 = 13 \leq c(16 - 4) \Leftrightarrow c \geq 13/12$
  - $T(5) = 9T(1) + 5 = 14 \leq c(25 - 5) \Leftrightarrow c \geq 14/20$
  - $T(6) = 9T(2) + 6$
- Non è necessario andare oltre, perchè  $T(6)$  dipende da  $T(2)$  che è già stato dimostrato

together now!

$$T(n) = \begin{cases} 9T(\lfloor n/3 \rfloor) + n & n > 1 \\ 1 & n \leq 1 \end{cases} \quad \text{Tentativo: } T(n) = O(n^2)$$

$$\exists c > 0, \exists m \geq 0 : T(n) \leq cn^2, \forall n \geq m$$

- Parametri:

- $c \geq \max \left\{ \frac{1}{2}, \frac{11}{2}, \frac{12}{6}, \frac{13}{12}, \frac{14}{20} \right\}$
  - $m = 2$

- Notare che l'esempio combina le due difficoltà insieme, ma è artificiale:

- Se avessimo scelto come ipotesi più stretta  $T(n) \leq cn^2 - bn$ , il problema sui casi base non si sarebbe posto

# Riassumendo

## Metodo di sostituzione

- Si “**indovina**” una possibile soluzione e si formula un’ipotesi induttiva
- Si **sostituisce** nella ricorrenza le espressioni  $T(\cdot)$ , utilizzando l’ipotesi induttiva
- Si **dimostra** che la soluzione è valida anche per il caso base

## Attenzione

- Ad ipotizzare soluzioni troppo “strette”
- Ad alcuni casi particolari che richiedono alcune astuzie matematiche
- Attenzione ai casi base: il logaritmo può complicare le cose

# Algoritmi e Strutture Dati

Analisi di algoritmi  
Ricorrenze comuni

Alberto Montresor

Università di Trento

2020/09/23

This work is licensed under a Creative Commons  
Attribution-ShareAlike 4.0 International License.



# Metodo dell'esperto

## Metodi per risolvere ricorrenze

- Metodo dell'albero di ricorsione, o per livelli
- Metodo di sostituzione, o per tentativi
- **Metodo dell'esperto, o delle ricorrenze comuni**

## Ricorrenze comuni

Esiste un'ampia classe di ricorrenze che possono essere risolte facilmente facendo ricorso ad alcuni teoremi, ognuno dei quali si occupa di una classe particolare di equazioni di ricorrenza.

# Ricorrenze lineari con partizione bilanciata

## Teorema

Siano  $a$  e  $b$  costanti intere tali che  $a \geq 1$  e  $b \geq 2$ , e  $c, \beta$  costanti reali tali che  $c > 0$  e  $\beta \geq 0$ . Sia  $T(n)$  data dalla relazione di ricorrenza:

$$T(n) = \begin{cases} aT(n/b) + cn^\beta & n > 1 \\ d & n \leq 1 \end{cases}$$

Posto  $\alpha = \log a / \log b = \log_b a$ , allora:

$$T(n) = \begin{cases} \Theta(n^\alpha) & \alpha > \beta \\ \Theta(n^\alpha \log n) & \alpha = \beta \\ \Theta(n^\beta) & \alpha < \beta \end{cases}$$

# Ricorrenze lineari con partizione bilanciata

## Assunzioni

Assumiamo che  $n$  sia una potenza intera di  $b$ :  $n = b^k, k = \log_b n$

## Perchè ci serve?

Semplifica tutti i calcoli successivi

## Influisce sul risultato?

- Supponiamo che l'input abbia dimensione  $b^k + 1$
- Estendiamo l'input fino ad una dimensione  $b^{k+1}$  (**padding**)
- L'input è stato esteso al massimo di un fattore costante  $b$
- Ininfluenza al fine della complessità computazionale



# Ricorrenze lineari con partizione bilanciata

$$T(n) = aT(n/b) + cn^\beta \quad T(1) = d$$

Liv.	Dim.	Costo chiam.	N. chiamate	Costo livello
0	$b^k$	$cb^{k\beta}$	1	$cb^{k\beta}$
1	$b^{k-1}$	$cb^{(k-1)\beta}$	$a$	$acb^{(k-1)\beta}$
2	$b^{k-2}$	$cb^{(k-2)\beta}$	$a^2$	$a^2cb^{(k-2)\beta}$
...	...	...	...	...
$i$	$b^{k-i}$	$cb^{(k-i)\beta}$	$a^i$	$a^i cb^{(k-i)\beta}$
...	...	...	...	...
$k-1$	$b$	$cb^\beta$	$a^{k-1}$	$a^{k-1}cb^\beta$
$k$	1	$d$	$a^k$	$da^k$

# Ricorrenze lineari con partizione bilanciata

Liv.	Dim.	Costo chiam.	N. chiamate	Costo livello
$i$	$b^{k-i}$	$cb^{(k-i)\beta}$	$a^i$	$a^i cb^{(k-i)\beta}$
$k$	1	$d$	$a^k$	$da^k$

Sommando i costi totali di tutti i livelli, si ottiene:

$$T(n) = da^k + cb^{k\beta} \sum_{i=0}^{k-1} \frac{a^i}{b^{i\beta}} = da^k + cb^{k\beta} \sum_{i=0}^{k-1} \left( \frac{a}{b^\beta} \right)^i$$

# Ricorrenze lineari con partizione bilanciata

$$T(n) = da^k + cb^{k\beta} \sum_{i=0}^{k-1} \frac{a^i}{b^{i\beta}} = da^k + cb^{k\beta} \sum_{i=0}^{k-1} \left( \frac{a}{b^\beta} \right)^i$$

## Osservazioni

- $a^k = a^{\log_b n} = a^{\log n / \log b} = 2^{\log a \log n / \log b} = n^{\log a / \log b} = n^\alpha$
- $\alpha = \log a / \log b \Rightarrow \alpha \log b = \log a \Rightarrow \log b^\alpha = \log a \Rightarrow a = b^\alpha$
- Poniamo  $q = \frac{a}{b^\beta} = \frac{b^\alpha}{b^\beta} = b^{\alpha-\beta}$

$$T(n) = da^k + cb^{k\beta} \sum_{i=0}^{k-1} \left( \frac{a}{b^\beta} \right)^i = dn^\alpha + cb^{k\beta} \sum_{i=0}^{k-1} q^i$$

# Ricorrenze lineari con partizione bilanciata

Caso 1:  $\alpha > \beta$

Ne segue che:  $q = b^{\alpha-\beta} > 1$ :

$$\begin{aligned}
 T(n) &= dn^\alpha + cb^{k\beta} \sum_{i=0}^{k-1} q^i && \text{Serie geometrica finita} \\
 &= n^\alpha d + cb^{k\beta} [(q^k - 1)/(q - 1)] && \text{Disequazione} \\
 &\leq n^\alpha d + cb^{k\beta} q^k / (q - 1) && \text{Sostituzione } q \\
 &= n^\alpha d + \frac{cb^{k\beta} a^k}{b^{k\beta}} / (q - 1) && \text{Passi algebrici} \\
 &= n^\alpha d + ca^k / (q - 1) && a^k = n^\alpha, \text{raccolta termini} \\
 &= n^\alpha [d + c/(q - 1)]
 \end{aligned}$$

- Quindi  $T(n)$  è  $O(n^\alpha)$ .
- Per via della componente  $dn^\alpha$ ,  $T(n)$  è anche  $\Omega(n^\alpha)$ , e quindi  $T(n) = \Theta(n^\alpha)$ .

# Ricorrenze lineari con partizione bilanciata

Caso 2:  $\alpha = \beta$

Ne segue che:  $q = b^{\alpha-\beta} = 1$ :

$$T(n) = dn^{\alpha} + cb^{k\beta} \sum_{i=0}^{k-1} q^i$$

$$= n^{\alpha}d + cn^{\beta}k$$

$$= n^{\alpha}d + cn^{\alpha}k$$

$$= n^{\alpha}(d + ck)$$

$$= n^{\alpha}[d + c \log n / \log b]$$

$$q^i = 1^i = 1$$

$$\alpha = \beta$$

Raccolta termini

$$k = \log_b n$$

e quindi  $T(n)$  è  $\Theta(n^{\alpha} \log n)$ ;

# Ricorrenze lineari con partizione bilanciata

## Caso 3: $\alpha < \beta$

Ne segue che:  $q = b^{\alpha-\beta} < 1$ :

$$\begin{aligned}
 T(n) &= dn^{\alpha} + cb^{k\beta} \sum_{i=0}^{k-1} q^i \\
 &= n^{\alpha}d + cb^{k\beta} [(q^k - 1)/(q - 1)] && \text{Serie geometrica finita} \\
 &= n^{\alpha}d + cb^{k\beta} [(1 - q^k)/(1 - q)] && \text{Inversione} \\
 &\leq n^{\alpha}d + cb^{k\beta} [1/(1 - q)] && \text{Diseguazione} \\
 &= n^{\alpha}d + c\textcolor{red}{n}^{\beta}/(1 - q) && b^k = n
 \end{aligned}$$

- Quindi  $T(n)$  è  $O(n^{\beta})$ .
- Poichè  $T(n) = \Omega(n^{\beta})$  per il termine non ricorsivo, si ha che  $T(n) = \Theta(n^{\beta})$ .

# Ricorrenze lineari con partizione bilanciata (Estesa)

## Teorema

Sia  $a \geq 1$ ,  $b > 1$ ,  $f(n)$  asintoticamente positiva, e sia

$$T(n) = \begin{cases} aT(n/b) + f(n) & n > 1 \\ d & n \leq 1 \end{cases}$$

Sono dati tre casi:

(1)	$\exists \epsilon > 0 : f(n) = O(n^{\log_b a - \epsilon})$	$\Rightarrow T(n) = \Theta(n^{\log_b a})$
(2)	$f(n) = \Theta(n^{\log_b a})$	$\Rightarrow T(n) = \Theta(f(n) \log n)$
(3)	$\exists \epsilon > 0 : f(n) = \Omega(n^{\log_b a + \epsilon}) \wedge$ $\exists c : 0 < c < 1, \exists m \geq 0 :$ $af(n/b) \leq cf(n), \forall n \geq m$	$\Rightarrow T(n) = \Theta(f(n))$

## Alcuni esempi

Ricorrenza	a	b	$\log_b a$	Caso	Funzione
$T(n) = 9T(n/3) + n \log n$					



## Alcuni esempi

Ricorrenza	a	b	$\log_b a$	Caso	Funzione
$T(n) = 9T(n/3) + n \log n$	9	3	2	(1)	$T(n) = \Theta(n^2)$

$$f(n) = n \log n = O(n^{\log_b a - \epsilon}) = O(n^{2 - \epsilon}), \text{ con } \epsilon < 1$$

## Alcuni esempi

Ricorrenza	a	b	$\log_b a$	Caso	Funzione
$T(n) = T(2n/3) + 1$					

## Alcuni esempi

Ricorrenza	a	b	$\log_b a$	Caso	Funzione
$T(n) = T(2n/3) + 1$	1	$\frac{3}{2}$	0	(2)	$T(n) = \Theta(\log n)$

$$f(n) = n^0 = \Theta(n^{\log_b a}) = \Theta(n^0)$$

## Alcuni esempi

Ricorrenza	a	b	$\log_b a$	Caso	Funzione
$T(n) = 3T(n/4) + n \log n$					

# Alcuni esempi

Ricorrenza	a	b	$\log_b a$	Caso	Funzione
$T(n) = 3T(n/4) + n \log n$	3	4	$\approx 0.79$	(3)	$T(n) = \Theta(n \log n)$

$$f(n) = n \log n = \Omega(n^{\log_4 3 + \epsilon}), \text{ con } \epsilon < 1 - \log_4 3 \approx 0.208$$

Dobbiamo dimostrare che:

$$\exists c \leq 1, \exists m \geq 0 : af(n/b) \leq cf(n), \forall n \geq m$$

$$\begin{aligned}
 af(n/b) &= 3(n/4 \log n/4) \\
 &= 3/4n(\log n - \log 4) \\
 &\leq 3/4n \log n \\
 &\stackrel{?}{\leq} cn \log n
 \end{aligned}$$

L'ultima disequazione è soddisfatta da  $c = 3/4$  e qualsiasi  $m$ .

## Alcuni esempi

Ricorrenza	a	b	$\log_b a$	Caso	Funzione
$T(n) = 2T(n/2) + n \log n$					

## Alcuni esempi

Ricorrenza	a	b	$\log_b a$	Caso	Funzione
$T(n) = 2T(n/2) + n \log n$	2	2	1	–	Non applicabile

$$f(n) = n \log n \neq O(n^{1-\epsilon}), \text{ con } \epsilon > 0$$

$$f(n) = n \log n \neq \Theta(n)$$

$$f(n) = n \log n \neq \Omega(n^{1+\epsilon}), \text{ con } \epsilon > 0$$

Nessuno dei tre casi è applicabile e bisogna utilizzare altri metodi.

# Interi inferiori/superiori

- I teoremi appena visti non considerano equazioni di ricorrenza con interi inferiori/superiori

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

- In realtà, le equazioni di ricorrenza dovrebbero sempre essere espresse tramite interi inferiori/superiori, perché operano su dimensioni dell'input

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n$$

- I risultati dei teoremi valgono anche quando le funzioni sono espresse tramite interi inferiori/superiori



# Ricorrenze lineari di ordine costante

## Teorema

Siano  $a_1, a_2, \dots, a_h$  costanti intere non negative, con  $h$  costante positiva,  $c$  e  $\beta$  costanti reali tali che  $c > 0$  e  $\beta \geq 0$ , e sia  $T(n)$  definita dalla relazione di ricorrenza:

$$T(n) = \begin{cases} \sum_{1 \leq i \leq h} a_i T(n-i) + cn^\beta & n > m \\ \Theta(1) & n \leq m \leq h \end{cases}$$

Posto  $a = \sum_{1 \leq i \leq h} a_i$ , allora:

- ❶  $T(n)$  è  $\Theta(n^{\beta+1})$ , se  $a = 1$ ,
- ❷  $T(n)$  è  $\Theta(a^n n^\beta)$ , se  $a \geq 2$ .

# Alcuni esempi

	<b>Ricorrenza</b>	<b>a</b>	$\beta$	<b>Caso</b>	<b>Funzione</b>
(A)	$T(n) = T(n - 10) + n^2$				
(B)	$T(n) = T(n - 2) + T(n - 1) + 1$				

## Alcuni esempi

	Ricorrenza	$a$	$\beta$	Caso	Funzione
(A)	$T(n) = T(n - 10) + n^2$	1	2	(1)	$T(n) = \Theta(n^3)$
(B)	$T(n) = T(n - 2) + T(n - 1) + 1$	2	0	(2)	$T(n) = 2^n$

(A) Poiché  $a = 1$ , il costo è polinomiale.

(B) Poiché  $a = 2$ , il costo è esponenziale.

## Esercizio

Siano

- $T(n) = 7T(n/2) + n^2$  una funzione di costo di un algoritmo  $A$ , e
- $T'(n) = aT'(n/4) + n^2$  una funzione di costo di un algoritmo  $A'$ .

Qual è il massimo valore intero di  $a$  che rende  $A'$  asintoticamente più veloce di  $A$ ?

## Esercizio – Soluzione

Poichè  $\log_2 7$  è  $\approx 2.81$ , il Master Theorem dice che  $T(n) = \Theta(n^{\log_2 7})$ .

Utilizzando alcune trasformazioni algebriche, si ottiene che:

$$\begin{aligned}\log_2 7 &= \frac{\log_4 7}{\log_4 2} = \frac{\log_4 7}{1/2} \\ &= 2 \log_4 7 = \log_4 49\end{aligned}$$

- $a < 16 \Rightarrow \alpha = \log_4 a < 2 \Rightarrow T'(n) = \Theta(n^2) = O(T(n))$
- $a = 16 \Rightarrow \alpha = \log_4 a = 2 \Rightarrow T'(n) = \Theta(n^2 \log n) = O(T(n))$
- $16 < a \leq 49 \Rightarrow \alpha = \log_4 a > 2 \Rightarrow T'(n) = \Theta(n^\alpha) = O(T(n))$
- $a < 49 \Rightarrow \alpha = \log_4 a > 2 \Rightarrow T'(n) = \Theta(n^\alpha) = \Omega(T(n))$

# Algoritmi e strutture dati

Analisi di funzioni  
Back to algorithms!

Alberto Montresor

Università di Trento

2020/09/23

This work is licensed under a Creative Commons  
Attribution-ShareAlike 4.0 International License.



# Sommario

- 1 Notazione asintotica
  - Definizioni
- 2 Proprietà della notazione asintotica
  - Funzioni di costo particolari
  - Proprietà delle notazioni
  - Altre funzioni di costo
  - Classificazione delle funzioni
- 3 Ricorrenze
  - Introduzione
  - Albero di ricorsione, o per livelli
  - Metodo della sostituzione
  - Metodo dell'esperto
- 4 Back to algorithms!
  - Ruolo dei fattori moltiplicativi

# Complessità della Versione 1

```

int maxsum1(int[] A, int n) {
    int maxSoFar = 0;
    for (int i=0; i < n; i++) {
        for (int j=i; j < n; j++) {
            int sum = 0;
            for (int k=i; k <= j; k++) {
                sum = sum + A[k];
            }
            maxSoFar = max(maxSoFar, sum);
        }
    }
    return maxSoFar;
}

```

La complessità dell'algoritmo può essere approssimata come segue (contando il numero di esecuzioni della riga più interna)

$$T(n) = \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} (j - i + 1)$$



# Complessità della Versione 1 - $O(n^3)$

Vogliamo provare che  $T(n) = O(n^3)$ , i.e.

$$\exists c_2 > 0, \exists m \geq 0 : T(n) \leq c_2 n^3, \forall n \geq m$$

$$\begin{aligned} T(n) &= \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} (j - i + 1) \\ &\leq \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} n \leq \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n \\ &= \sum_{i=0}^{n-1} n^2 = n^3 \leq c_2 n^3 \end{aligned}$$

Questa disequazione è vera per  $n \geq m = 0$  and  $c_2 \geq 1$ .

# Complessità della Versione 1 - $\Omega(n^3)$

Vogliamo provare che  $T(n) = \Omega(n^3)$ , i.e.

$$\exists c_1 > 0, \exists m \geq 0 : T(n) \geq c_1 n^3, \forall n \geq m$$

$$\begin{aligned} T(n) &= \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} (j - i + 1) \\ &\geq \sum_{i=0}^{n/2} \sum_{j=i}^{i+n/2-1} (j - i + 1) \\ &= \sum_{i=0}^{n/2} \sum_{j=i}^{i+n/2-1} n/2 \\ &= \sum_{i=0}^{n/2} n^2/4 \geq n^3/8 \geq c_1 n^3 \end{aligned}$$

L'ultima disequazione è vera per  $n \geq m = 0$  and  $c_1 \leq 8$ .

## Complessità della versione 2

```

int maxsum2(int[] A, int n) {
    int maxSoFar = 0;
    for (int i=0; i < n; i++) {
        int sum = 0;
        for (int j=i; j < n; j++) {
            sum = sum + A[j];
            maxSoFar = max(maxSoFar, sum);
        }
    }
    return maxSoFar;
}

```

La complessità di questo algoritmo può essere approssimata come segue (stiamo contando il numero di passi nel ciclo più interno)

$$T(n) = \sum_{i=0}^{n-1} n - i$$

## Complessità della versione 2 - $\theta(n^2)$

Vogliamo provare che  $T(n) = \theta(n^2)$ .

$$\begin{aligned} T(n) &= \sum_{i=0}^{n-1} n - i \\ &= \sum_{i=1}^n i \\ &= \frac{n(n+1)}{2} = \Theta(n^2) \end{aligned}$$

Questo non richiede ulteriori dimostrazioni

# Complessità della versione 3

```

int maxsum_rec(int[] A, int i, int j) {
    if (i==j)
        return max(0, A[i]);
    int m = (i+j) / 2;
    int maxs = maxsum_rec(A, i, m);
    int maxd = maxsum_rec(A, m+1, j);
    int maxss = 0;
    int sum = 0;
    for (int k=m; k>=i; k--) {
        sum = sum+A[k];
        maxss = max(maxss, sum);
    }
    int maxdd = 0;
    sum = 0;
    for (int k=m+1; k<=j; k++) {
        sum = sum+A[k];
        maxdd = max(maxdd, sum);
    }
    return max(max(maxs,maxd),maxss+maxdd);
}

```

Per questo, definiamo la equazione di ricorrenza:

# Complessità della versione 3

```

int maxsum_rec(int[] A, int i, int j) {
    if (i==j)
        return max(0, A[i]);
    int m = (i+j) / 2;
    int maxs = maxsum_rec(A, i, m);
    int maxd = maxsum_rec(A, m+1, j);
    int maxss = 0;
    int sum = 0;
    for (int k=m; k>=i; k--) {
        sum = sum+A[k];
        maxss = max(maxss, sum);
    }
    int maxdd = 0;
    sum = 0;
    for (int k=m+1; k<=j; k++) {
        sum = sum+A[k];
        maxdd = max(maxdd, sum);
    }
    return max(max(maxs,maxd),maxss+maxdd);
}

```

Per questo, definiamo la equazione di ricorrenza:

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

Utilizzando il teorema, possiamo vedere che  $\alpha = \log_2 2 = 1$  e  $\beta = 1$ , quindi  $T(n) = \Theta(n \log n)$ .

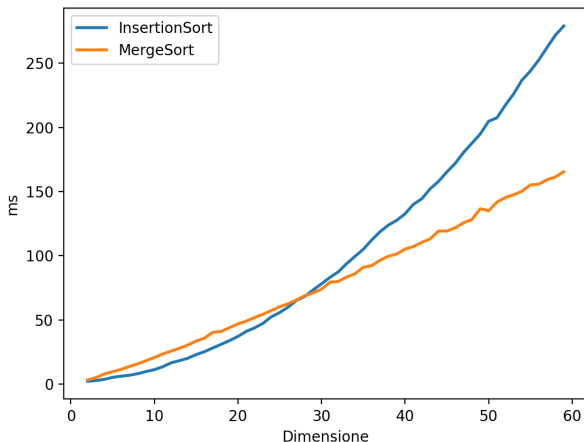
## Complessità della versione 4

```
int maxsum4(int A[], int n) {  
    int maxSoFar = 0;  
    int maxHere = 0;  
    for (int i=0; i < n; i++) {  
        maxHere = max(maxHere+A[i],0);  
        maxSoFar = max(maxSoFar,maxHere);  
    }  
    return maxSoFar;  
}
```

E' facile vedere che la complessità di questa versione è  $\theta(n)$ .

# Fattori moltiplicativi

A volte, i fattori moltiplicativi di una funzione di complessità sono talmente alti che se ne sconsiglia l'uso per piccoli valori di  $n$





# GNU Multiple Precision Arithmetic Library

- Utilizzata da Mathematica, Maple, etc.
- Le moltiplicazioni vengono realizzate utilizzando algoritmi diversi, mano a mano che  $n$  cresce.
- <https://gmplib.org/manual/Multiplication-Algorithms.html>

## 15.1 Multiplication

$N \times N$  limb multiplications and squares are done using one of seven algorithms, as the size  $N$  increases.

Algorithm Threshold

Basecase (none)

Karatsuba MUL\_TOOM22\_THRESHOLD

Toom-3 MUL\_TOOM33\_THRESHOLD

Toom-4 MUL\_TOOM44\_THRESHOLD

Toom-6.5 MUL\_TOOM6H\_THRESHOLD

Toom-8.5 MUL\_TOOM8H\_THRESHOLD

FFT MUL\_FFT\_THRESHOLD

# GNU Multiple Precision Arithmetic Library

- Utilizzata da Mathematica, Maple, etc.
- I limiti (threshold) dipendono dall'architettura

host type	abi	host name	meas thres	conf thres	cfg file
z10-ibm-linux-gnu	64	<a href="#">lgentoo4.s390.gentoo.wh0rd.org-stat</a>	1728	1728	<a href="#">s390_64/z10/gmp-mparam.h</a>
atom-unknown-linux-gnu	64	<a href="#">gege.gmpilib.org-stat</a>	2240	2240	<a href="#">x86_64/atom/gmp-mparam.h</a>
z10esa-ibm-linux-gnu	32	<a href="#">lgentoo3.s390.gentoo.wh0rd.org-stat</a>	2240	2240	<a href="#">s390_32/esame/gmp-mparam.h</a>
power7-unknown-linux-gnu	mode32	<a href="#">gcc1-power7.osuosl.org-stat</a>	2688	2688	<a href="#">powerpc64/mode32/p4/gmp-mparam.h</a>
bulldozer-unknown-freebsd8.3	64	<a href="#">oshell.gmpilib.org-stat</a>	3520	3712	<a href="#">x86_64/bd11/gmp-mparam.h</a>
piledriver-unknown-netbsd6.1.3	64	<a href="#">pilenbsd64v61.gmpilib.org-stat</a>	3712	3712	<a href="#">x86_64/bd2/gmp-mparam.h</a>
powerpc7447-unknown-linux-gnu	32	<a href="#">spigg.gmpilib.org-stat</a>	3712	3712	<a href="#">powerpc32/gmp-mparam.h</a>
coreihw1-unknown-netbsd6.1.2	64	<a href="#">hannahbsd64v61.gmpilib.org-stat</a>	4224	4224	<a href="#">x86_64/coreihw1/gmp-mparam.h</a>
coreinhm-unknown-netbsd6.1.3	64	<a href="#">hikonbsd64v61.gmpilib.org-stat</a>	4224	4032	<a href="#">x86_64/coreinhm/gmp-mparam.h</a>
power7-ibm-aix7.1.0.0	mode64	<a href="#">power-aix.fsffrance.org-stat</a>	4288	4288	<a href="#">powerpc64/mode64/p7/gmp-mparam.h</a>
atom-unknown-linux-gnu	32	<a href="#">gege.gmpilib.org-stat</a>	4544	4544	<a href="#">x86/atom/gmp-mparam.h</a>
core2-unknown-netbsd6.1.4	64	<a href="#">repentiumbsd64v61.gmpilib.org-stat</a>	4736	4736	<a href="#">x86_64/core2/gmp-mparam.h</a>
coreisbr-apple-darwin12.5.0	64	<a href="#">poire.loria.fr-stat</a>	4736	4736	<a href="#">x86_64/coreisbr/gmp-mparam.h</a>
coreiwsn-unknown-linux-gnu	64	<a href="#">gcc20.fsffrance.org-stat</a>	4736	4032	<a href="#">x86_64/coreinhm/gmp-mparam.h</a>
power7-unknown-linux-gnu	mode64	<a href="#">gcc1-power7.osuosl.org-stat</a>	4736	4288	<a href="#">powerpc64/mode64/p7/gmp-mparam.h</a>
powerpc970-apple-darwin8.11.0	mode32	<a href="#">g5.gmpilib.org-stat</a>	4736	2688	<a href="#">powerpc64/mode32/p4/gmp-mparam.h</a>
power7-ibm-aix7.1.0.0	32	<a href="#">power-aix.fsffrance.org-stat</a>	5312	5312	<a href="#">powerpc32/p7/gmp-mparam.h</a>
bobcat-unknown-netbsd6.1.3	64	<a href="#">bobcat.gmpilib.org-stat</a>	5504	5504	<a href="#">x86_64/bobcat/gmp-mparam.h</a>
alphaev6-unknown-linux-gnu	standard	<a href="#">agnesi.math.su.se-stat</a>	5760	5760	<a href="#">alpha/ev6/gmp-mparam.h</a>
armcortexa15neon-unknown-linux-	standard	<a href="#">parma.gmpilib.org-stat</a>	5760	5760	<a href="#">arm/v7a/cora15/gmp-mparam.h</a>
power7-unknown-linux-gnu	32	<a href="#">gcc1-power7.osuosl.org-stat</a>	5760	5312	<a href="#">powerpc32/p7/gmp-mparam.h</a>
core2-unknown-netbsdelf6.1.4	32	<a href="#">repentiumbsd32v61.gmpilib.org-stat</a>	6784	6784	<a href="#">x86/core2/gmp-mparam.h</a>
coreinhm-unknown-netbsdelf6.1.3	32	<a href="#">hikonbsd32v61.gmpilib.org-stat</a>	6784	6784	<a href="#">x86/coreinhm/gmp-mparam.h</a>