Progetto e analisi di algoritmi

Roberto Cordone

DTI - Università degli Studi di Milano

Polo Didattico e di Ricerca di Crema

Tel. 0373 / 898**089**

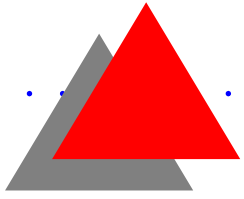
E-mail: **cordone@dti.unimi.it**

Ricevimento: **su appuntamento**

Web page: **http://www.dti.unimi.it/~cordone**

Lezioni: **Martedì dalle 11.00 alle 13.00**

Giovedì dalle 11.00 alle 13.00



Risorse computazionali

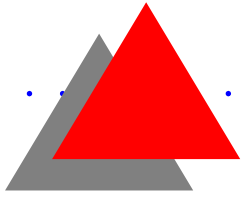
Quanto costa risolvere un problema P con un algoritmo A?

1. potenza: numero di stati di una *TM* che esegue l'algoritmo
2. tempo: numero di passi $T_A(I)$ compiuti prima dell'arresto
3. spazio: numero di celle $S_A(I)$ scritte prima dell'arresto

Le tre misure sono di natura differente

- la potenza è la stessa per tutte le istanze I ("costo fisso")
- tempo e spazio dipendono da I ("costo variabile")

Tempo e spazio sono misure di costo più significative



Complessità

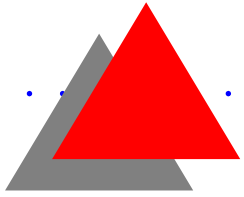
Più in generale (non solo usando *TM*)

- complessità temporale di un algoritmo A su un'istanza I il tempo $T_A(I)$ che A impiega a risolvere I
- complessità spaziale di un algoritmo A su un'istanza I lo spazio $S_A(I)$ che A usa per risolvere I

A seconda del modello, bisogna definire tempo e spazio

Problemi

- La complessità dipende dall'alfabeto usato
- La complessità dipende dal modello computazionale usato



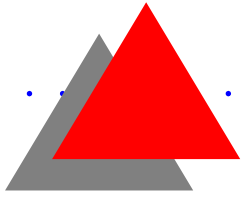
Complessità e alfabeti

Dimensione di un'istanza I è la lunghezza $|I|$ della stringa che la codifica (numero di occorrenze)

Per alfabeti $\mathcal{A}$ e $\mathcal{A}'$ con $|\mathcal{A}|$ e $|\mathcal{A}'| \geq 2$, la dimensione I varia di un fattore costante $|I|_{\mathcal{A}'} = \lceil \lg_{|\mathcal{A}'|} |\mathcal{A}| \rceil \, |I|_{\mathcal{A}}$

$\Rightarrow$ La complessità dipende dall'alfabeto, ma varia di un fattore costante al variare dell'alfabeto

Che succede con l'alfabeto unario? ($|I|$ lunghissima $\Rightarrow \ldots$)
E con "alfabeti" infiniti, come $\mathbb{N}$? ($|I| = 1 \Rightarrow \ldots$)



Equivalenza polinomiale

Modelli (polinomialmente) equivalenti: risolvono gli stessi problemi (in tempi e spazi legati da un opportuno polinomio)

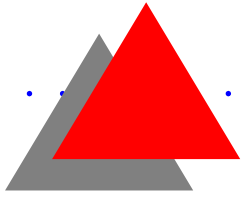
I modelli equivalenti alla TM le sono polinomialmente equivalenti!

Se esiste una macchina *RAM* che risolve ogni istanza I di un problema P in tempo $T(I)$ e spazio $S(I)\ldots$

$\ldots$ allora, esistono due polinomi $p(\cdot)$ e $q(\cdot)$ e una *TM* che risolve I in tempo $T' \leq p(T(I))$ e spazio $S' \leq q(S(I))$

Esempio (stringhe palindrome)

$$T_{RAM}(I) = c|I| \qquad T_{TM}(I) \leq p(T_{RAM}(I)) = p(c|I|)$$



Algoritmi polinomiali

Algoritmo polinomiale è un algoritmo la cui complessità è un polinomio π della dimensione dell'istanza: $T(I) = \pi(|I|)$

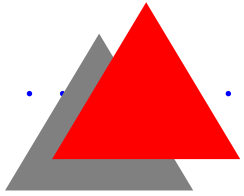
Ma lo spazio dei polinomi è chiuso rispetto alla composizione

$$p(\cdot) \text{ e } \pi(\cdot) \text{ sono polinomi } \Rightarrow p(\pi(\cdot)) \text{ è un polinomio}$$

Un algoritmo (non) polinomiale su una macchina è (non) polinomiale su tutte le macchine polinomialmente equivalenti

La polinomialità non dipende dalla macchina!

Possiamo ragionare ogni volta su quella che preferiamo



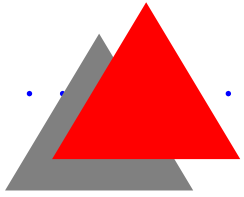
Confronto fra algoritmi (1)

Sappiamo confrontare due algoritmi A e A' su un'istanza I:
A è meglio di A' quando impiega meno tempo (spazio)

$$A(I) \preceq A'(I) \Leftrightarrow T_A(I) \leq T_{A'}(I)$$

Ora vogliamo confrontarli sull'intero problema P, cioè stabilire una relazione di ordine debole fra algoritmi per un problema:
per ogni terna di algoritmi A, A' e A'' che risolvono P

1. (riflessività) $A \preceq A$
2. (transitività) $A \preceq A'$ e $A' \preceq A'' \Rightarrow A \preceq A''$
3. (completezza) vale almeno una fra $A \preceq A'$ e $A' \preceq A$



Confronto fra algoritmi (2)

Consideriamo quattro possibili definizioni

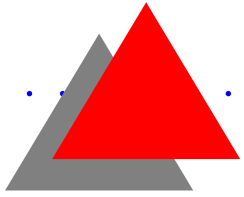
- su tutte le istanze? *Rarissimo...*
- nel caso migliore? *E se non siamo fortunati...?*
- in media? *Ma quale media...?*
- nel caso peggiore? *Perché essere pessimisti...?*

Def. 1: Complessità su tutte le istanze

$$A \preceq A' \quad \Leftrightarrow \quad T_A(I) \leq T_{A'}(I)\ \forall I \in P$$

Complicata da verificare e l'ordine non è quasi mai completo

(per alcune istanze è meglio A, per altre A')



Confronto fra algoritmi (3)

Def. 2: Complessità nel caso migliore

$$A \preceq A' \quad \Leftrightarrow \quad \min_{I \in P} T_A(I) \leq \min_{I \in P} T_{A'}(I)$$

Ogni problema ha istanze banali, ma sono poco significative!

Def. 3: Complessità nel caso medio

$$A \preceq A' \quad \Leftrightarrow \quad E\left[T_A(I)\right] \leq E\left[T_{A'}(I)\right]$$

- richiede di considerare tutte le istanze
- richiede la distribuzione di probabilità delle istanze
- richiede calcoli complicati

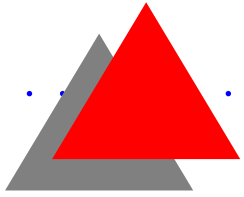
Sarà possibile e utile solo in casi limitati

Confronto fra algoritmi (4)

Def. 4: Complessità nel caso peggiore

- Spesso è facile identificare le istanze peggiori
- Fornisce un limite superiore,
  che è un'informazione comunque utile
- Spesso il caso peggiore è molto frequente
  (ad es., l'insuccesso in una ricerca)
- Spesso la complessità nel caso peggiore è simile a quella nel caso medio

È una definizione sbilanciata, ma pratica e utile



Complessità nel caso peggiore

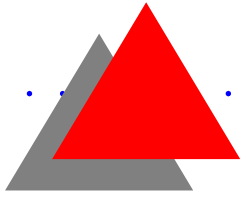
Ma che significa “caso peggiore”?

$\nexists \max_{I \in P} T_A(I)$ perché di solito $\sup_{I \in P} T_A(I) = +\infty$

Per dare senso alla definizione, ragioniamo

- a dimensione fissata $P_n = \{I \in P : |I| = n\}$
- nel caso peggiore

$$T_A(n) = \max_{I \in P_n} T_A(I)$$



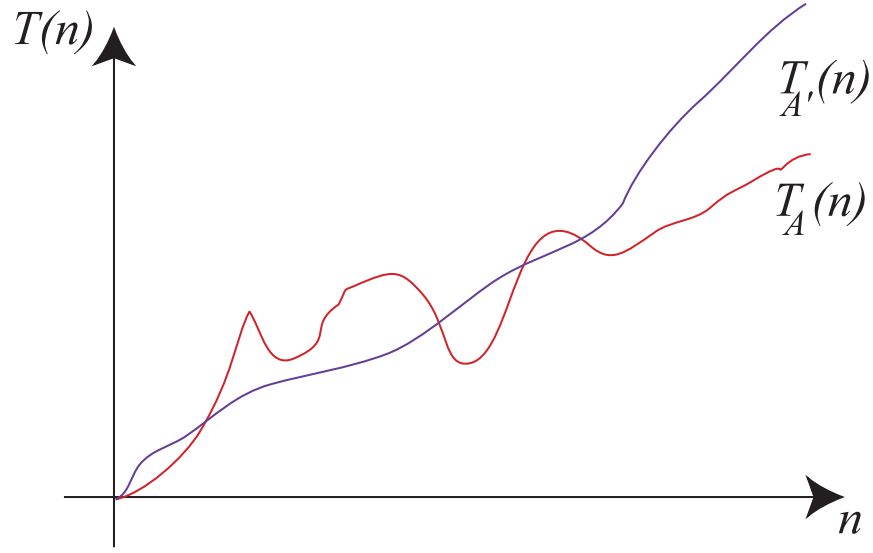
Complessità e dimensione

Ma che significa “caso peggiore”?

La complessità cresce indefinitamente con n...

- considerare tutti i valori di n? *Ci risiamo...*
- considerare valori di n sufficientemente grandi?

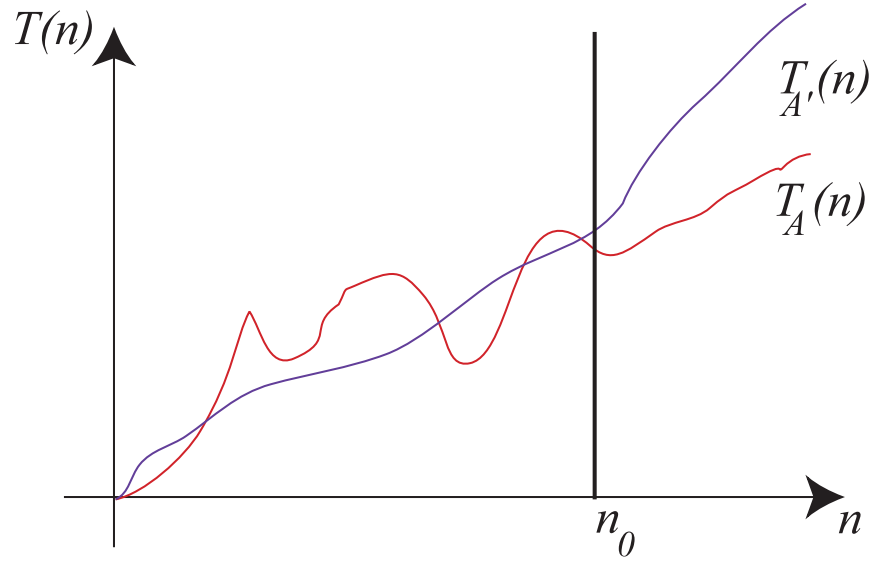
Quanto grandi...?



Complessità asintotica (1)

Un algoritmo A è meglio di un algoritmo A' quando A usa meno risorse di A'

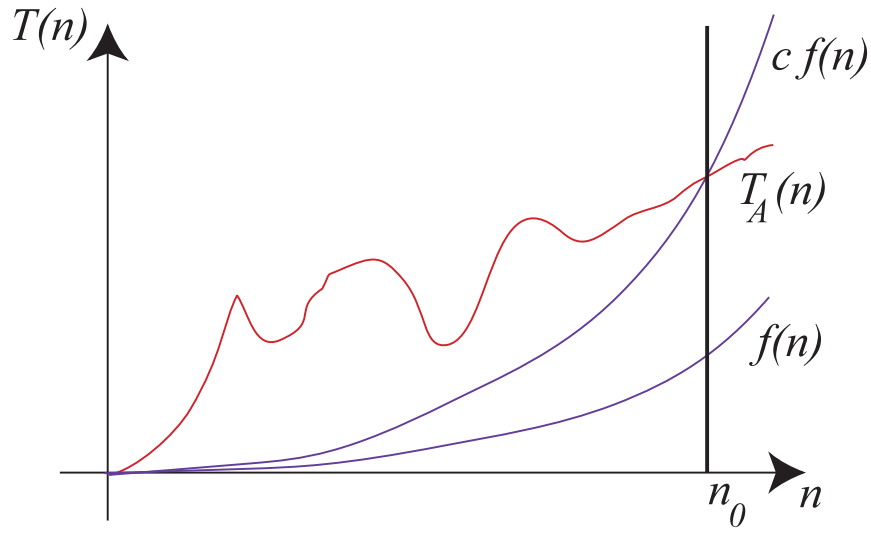
- sulla peggior istanza di dimensione n
- per ogni valore di $n \geq n_0$



Complessità asintotica (2)

$$T(n) \in O(f(n))$$

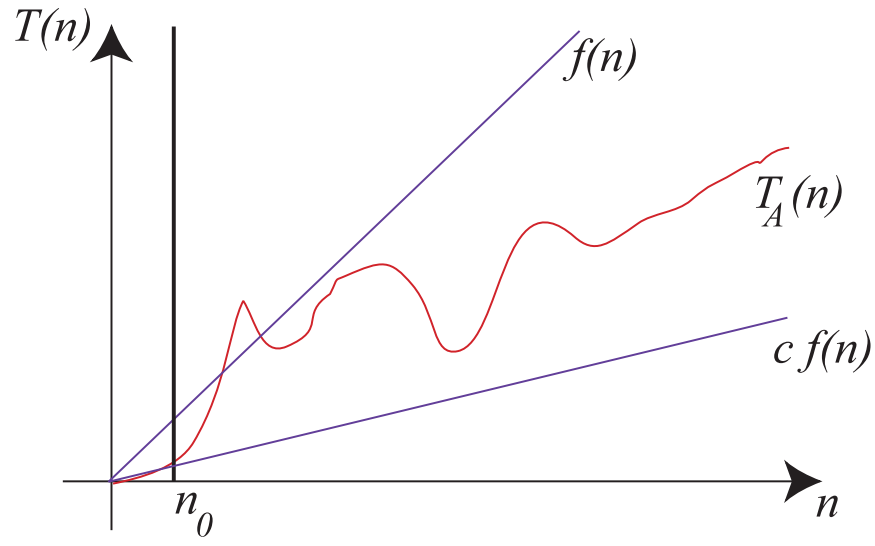
$$\exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad T(n) \leq cf(n) \quad \forall n \geq n_0$$



Complessità asintotica (3)

$$T(n) \in \Omega(f(n))$$

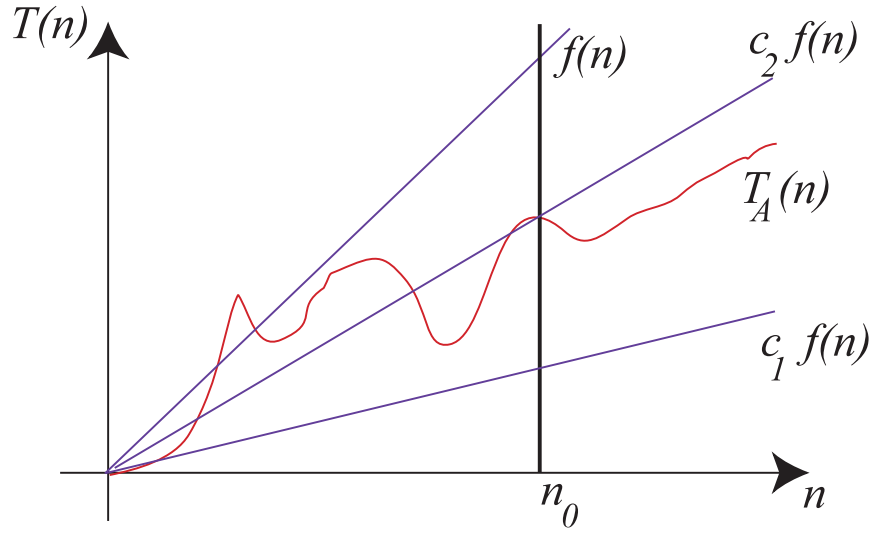
$$\exists c \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad T(n) \geq cf(n) \geq 0 \quad \forall n \geq n_0$$



Complessità asintotica (4)

$$T(n) \in \Theta(f(n))$$

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad 0 \leq c_1 f(n) \leq T(n) \leq c_2 f(n) \quad \forall n \geq n_0$$

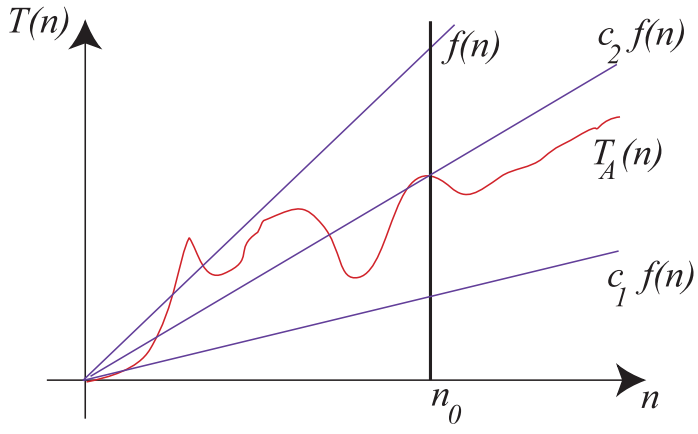


Più in dettaglio

$$T(n) \in \Theta(f(n))$$

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad 0 \leq c_1 f(n) \leq T(n) \leq c_2 f(n) \quad \forall n \geq n_0$$

$T(n)$ è chiusa *a sandwich* fra $c_1 f(n)$ e $c_2 f(n)$



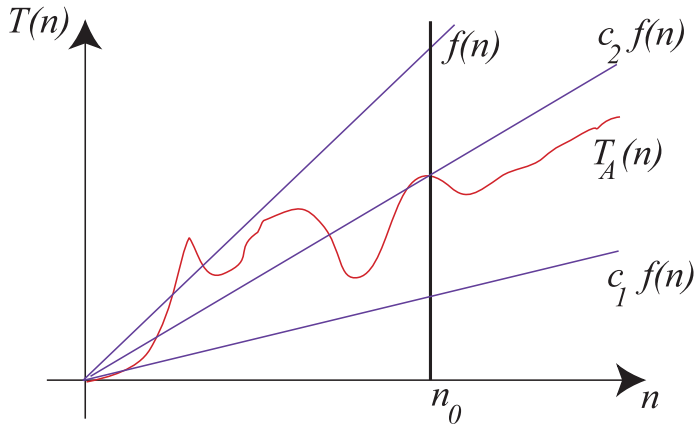
- per qualche valore *piccolo* di c_1
- per qualche valore *grosso* di c_2
- per ogni valore *abbastanza grosso* di n
- per qualche definizione di n *"abbastanza grosso"*

Più in dettaglio

$$T(n) \in \Theta(f(n))$$

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad 0 \leq c_1 f(n) \leq T(n) \leq c_2 f(n) \quad \forall n \geq n_0$$

$T(n)$ è chiusa *a sandwich* fra $c_1 f(n)$ e $c_2 f(n)$



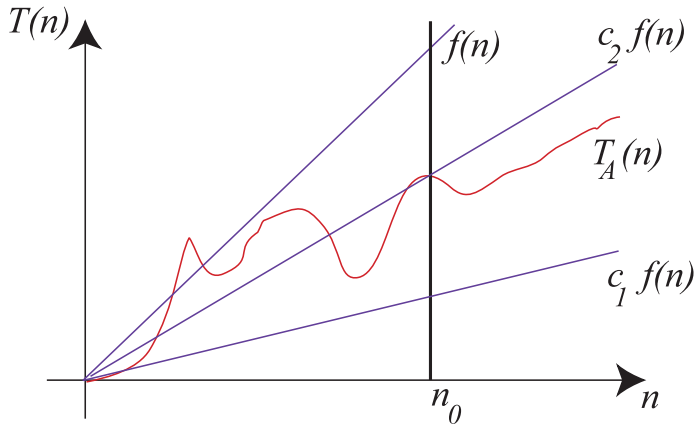
- per qualche valore *piccolo* di c_1
- per qualche valore *grosso* di c_2
- per ogni valore *abbastanza grosso* di n
- per qualche definizione di n “*abbastanza grosso*”

Più in dettaglio

$$T(n) \in \Theta(f(n))$$

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad 0 \leq c_1 f(n) \leq T(n) \leq c_2 f(n) \quad \forall n \geq n_0$$

$T(n)$ è chiusa *a sandwich* fra $c_1 f(n)$ e $c_2 f(n)$



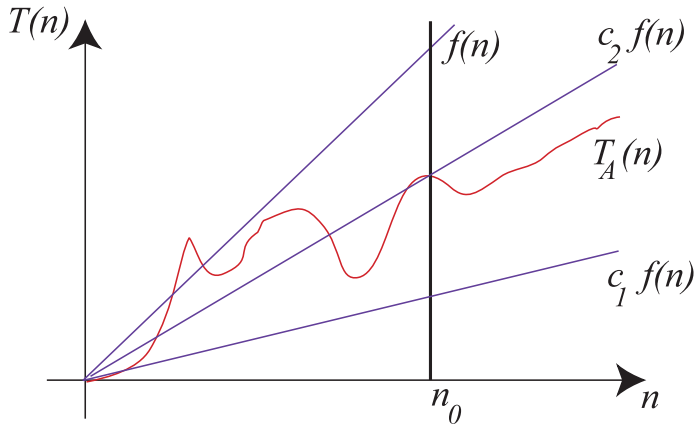
- per qualche valore *piccolo* di c_1
- per qualche valore *grosso* di c_2
- per ogni valore *abbastanza grosso* di n
- per qualche definizione di n *"abbastanza grosso"*

Più in dettaglio

$$T(n) \in \Theta(f(n))$$

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad 0 \leq c_1 f(n) \leq T(n) \leq c_2 f(n) \quad \forall n \geq n_0$$

$T(n)$ è chiusa *a sandwich* fra $c_1 f(n)$ e $c_2 f(n)$



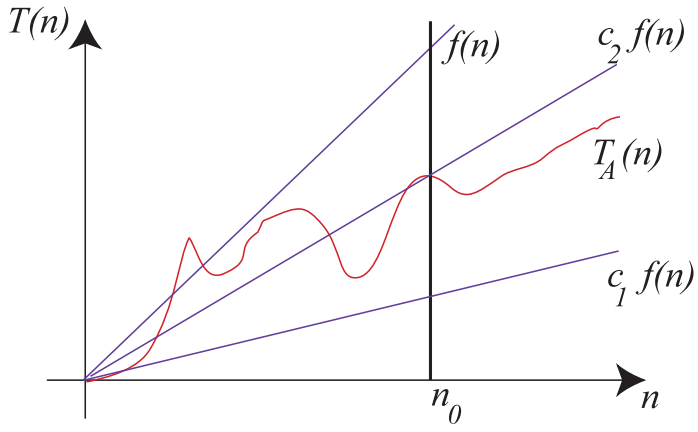
- per qualche valore *piccolo* di c_1
- per qualche valore *grosso* di c_2
- per ogni valore *abbastanza grosso* di n
- per qualche definizione di n *"abbastanza grosso"*

Più in dettaglio

$$T(n) \in \Theta(f(n))$$

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad 0 \leq c_1 f(n) \leq T(n) \leq c_2 f(n) \quad \forall n \geq n_0$$

$T(n)$ è chiusa *a sandwich* fra $c_1 f(n)$ e $c_2 f(n)$



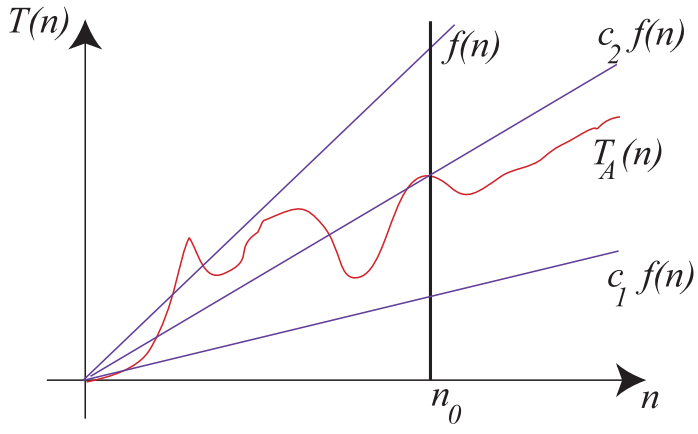
- per qualche valore *piccolo* di c_1
- per qualche valore *grosso* di c_2
- per ogni valore *abbastanza grosso* di n
- per qualche definizione di n *"abbastanza grosso"*

Più in dettaglio

$$T(n) \in \Theta(f(n))$$

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : \quad 0 \leq c_1 f(n) \leq T(n) \leq c_2 f(n) \quad \forall n \geq n_0$$

$T(n)$ è chiusa *a sandwich* fra $c_1 f(n)$ e $c_2 f(n)$



- per qualche valore *piccolo* di c_1
- per qualche valore *grosso* di c_2
- per ogni valore *abbastanza grosso* di n
- per qualche definizione di n *"abbastanza grosso"*

Esercizio 1

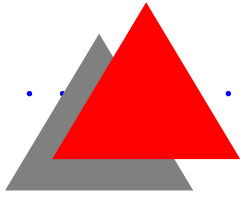
Dimostrare che $T(n) = 3n^2 + 7n + 8 \in \Theta(n^2)$

$$\exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ :$$
$$c_1 n^2 \leq 3n^2 + 7n + 8 \leq c_2 n^2 \quad \forall n \geq n_0$$

Per intuizione o per tentativi, sia $c_1 = 3$, $c_2 = 4$

- La prima disuguaglianza è vera per ogni $n \geq 1$
- La seconda disuguaglianza è vera per $7n + 8 \leq n^2 \Rightarrow$ per ogni $n \geq 8$

$c_1 = 3$, $c_2 = 4$ e $n_0 = 8$ soddisfano la definizione



Esercizio 2

Dimostrare che $T(n) = 3n^2 - 7n + 2 \notin \Theta(n)$

$$\nexists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}^+ : c_1 n \leq 3n^2 - 7n + 2 \leq c_2 n \quad \forall n \geq n_0$$

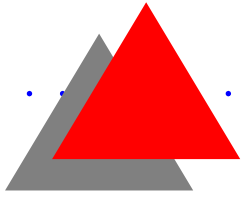
$\exists n \geq n_0 : c_1 n > 3n^2 - 7n + 2$o $3n^2 - 7n + 2 > c_2 n \forall c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N}$

Scegliamo la seconda disuguaglianza (ne basta una!)

Se $\phi(c_2) = (7 + c_2)^2 - 24 < 0$, è verificata $\forall n \in \mathbb{N}$. Altrimenti, lo è per $n < \frac{7+c_2-\sqrt{\phi(c_2)}}{6}$ o $n > \frac{7+c_2+\sqrt{\phi(c_2)}}{6}$. Si può dimostrare la seconda disuguaglianza, e insieme anche $n \geq n_0$, ponendo

$$n = \max\left(n_0, \left\lceil \frac{7 + c_2 + \sqrt{\phi(c_2)}}{6} \right\rceil + 1\right).$$

Si noti che qui n dipende da c_1, c_2 e n_0



Altri esercizi

Mostrare che:

- $n^2 \in \Theta\left(n^2 + 4n + 3\right)$
- $4n^2 \notin \Theta\left(n^3\right)$
- $n^2 \in \Omega\left(n^2 + 2n + 5\right)$
- $n^2 \in O\left(n^2/4 - 2\right)$
- $n^2 \notin O\left(1\,000\,000n\right)$
- $\sum_{i=1}^{k} n^i \in \Theta\left(n^k\right)$
- $\sum_{i=1}^{n} i^k \in O\left(n^{k+1}\right)$

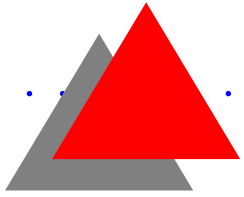
Proprietà fondamentali (1)

Sono facili da ricordare, se si associano

- $O \leftrightarrow \leq$
- $\Omega \leftrightarrow \geq$
- $\Theta \leftrightarrow =$

Riflessività

- $f(n) \in O(f(n))$
- $f(n) \in \Omega(f(n))$
- $f(n) \in \Theta(f(n))$



Proprietà fondamentali (2)

Simmetria trasposta (solo per O e Ω)

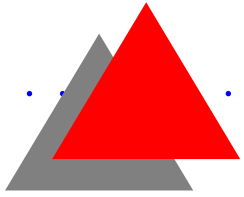
- $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$
- $f(n) \in \Omega(g(n)) \Leftrightarrow g(n) \in O(f(n))$

Simmetria (solo per Θ)

- $f(n) \in \Theta(g(n)) \Leftrightarrow g(n) \in \Theta(f(n))$

Antisimmetria

- $\begin{cases} T(n) \in O(f(n)) \\ T(n) \in \Omega(f(n)) \end{cases} \Leftrightarrow T(n) \in \Theta(f(n))$

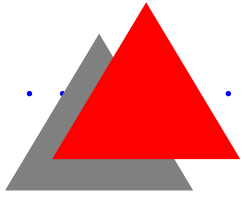


Proprietà fondamentali (3)

Transitività

- $$\begin{cases} f(n) \in O(g(n)) \\ g(n) \in O(h(n)) \end{cases} \Rightarrow f(n) \in O(h(n))$$
- $$\begin{cases} f(n) \in \Omega(g(n)) \\ g(n) \in \Omega(h(n)) \end{cases} \Rightarrow f(n) \in \Omega(h(n))$$
- $$\begin{cases} f(n) \in \Theta(g(n)) \\ g(n) \in \Theta(h(n)) \end{cases} \Rightarrow f(n) \in \Theta(h(n))$$

Esercizio: dimostrare queste proprietà...



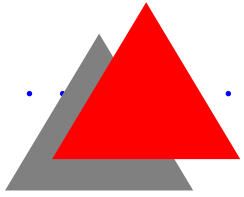
Proprietà fondamentali (4)

Date le funzioni $f(n)$ e $g(n)$ è possibile che

- $f(n) \notin O(g(n))$
- $f(n) \notin \Omega(g(n))$
- $f(n) \notin \Theta(g(n))$

Non vale la completezza!

(che invece vale per $\leq$, $\geq$ e $=$)

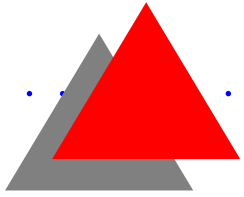


Ordinamento sugli algoritmi

Stiamo cercando di stabilire un ordinamento debole fra gli algoritmi che risolvono un problema dato

- $A \preceq A'$ significa
  $T_A(n) \in O(T_{A'}(n)) \Leftrightarrow T_{A'}(n) \in \Omega(T_A(n))$
- $A \succeq A'$ significa
  $T_A(n) \in \Omega(T_{A'}(n)) \Leftrightarrow T_{A'}(n) \in O(T_A(n))$
- $A \equiv A'$ significa
  $T_A(n) \in \Theta(T_{A'}(n)) \Leftrightarrow T_{A'}(n) \in \Theta(T_A(n))$

Tutte le proprietà sopra elencate si trasmettono dalle funzioni complessità agli algoritmi



Completezza

La completezza vale quasi sempre, ma non sempre!

Ma le coppie di funzioni per cui non vale sono piuttosto strane...

$$f_P(n) = \begin{cases} 2^{n^2} \text{ per } n \text{ pari} \\ 2^{(n-1)^2} \text{ per } n \text{ dispari} \end{cases} \qquad f_D(n) \begin{cases} 2^{(n-1)^2} \text{ per } n \text{ pari e} \geq 2 \\ 1 \text{ per } n = 0 \\ 2^{n^2} \text{ per } n \text{ dispari} \end{cases}$$

Per valori pari di n, $\dfrac{f_P(n)}{f_D(n)} = 2^{2n-1} \Rightarrow f_P(n) \notin O(f_D)$

Per valori dispari di n, $\dfrac{f_P(n)}{f_D(n)} = 2^{1-2n} \Rightarrow f_P(n) \notin \Omega(f_D)$

Abusi di notazione

- $f(n) = O(g(n))$ significa $f(n) \in O(g(n))$
- $n^2 + 3n + 5 = n^2 + O(n)$ significa

  $$\exists f(n) \in O(n) : n^2 + 3n + 5 = n^2 + f(n) \quad \forall n$$

  (nel nostro caso, $f(n) = 3n + 5$)
- $2n^2 + O(n) = O\left(n^2\right)$ significa

  $$\forall f_1(n) \in O(n)\, \exists f_2(n) \in O\left(n^2\right) : 2n^2 + f_1(n) = f_2(n) \quad \forall n$$

- Però $T(n) = \sum_{i=1}^{n} O(i)$ non significa

  $$\exists f_1(i) \in O(i), \ldots, f_n(i) \in O(i) : T(n) = f_1(1) + \ldots + f_n(n) \ \forall n$$

  bensì $\exists f(i) \in O(i) : T(n) = \sum_{i=1}^{n} f(i) \quad \forall n$