Algoritmi e Strutture Dati

Analisi di algoritmi

Maria Rita Di Berardini[2], Emanuela Merelli[1]

[1]Dipartimento di Matematica e Informatica
Università di Camerino

[2]Polo di Scienze
Università di Camerino ad Ascoli Piceno

Parte I

Analisi asintotica

Un graduale processo di astrazione

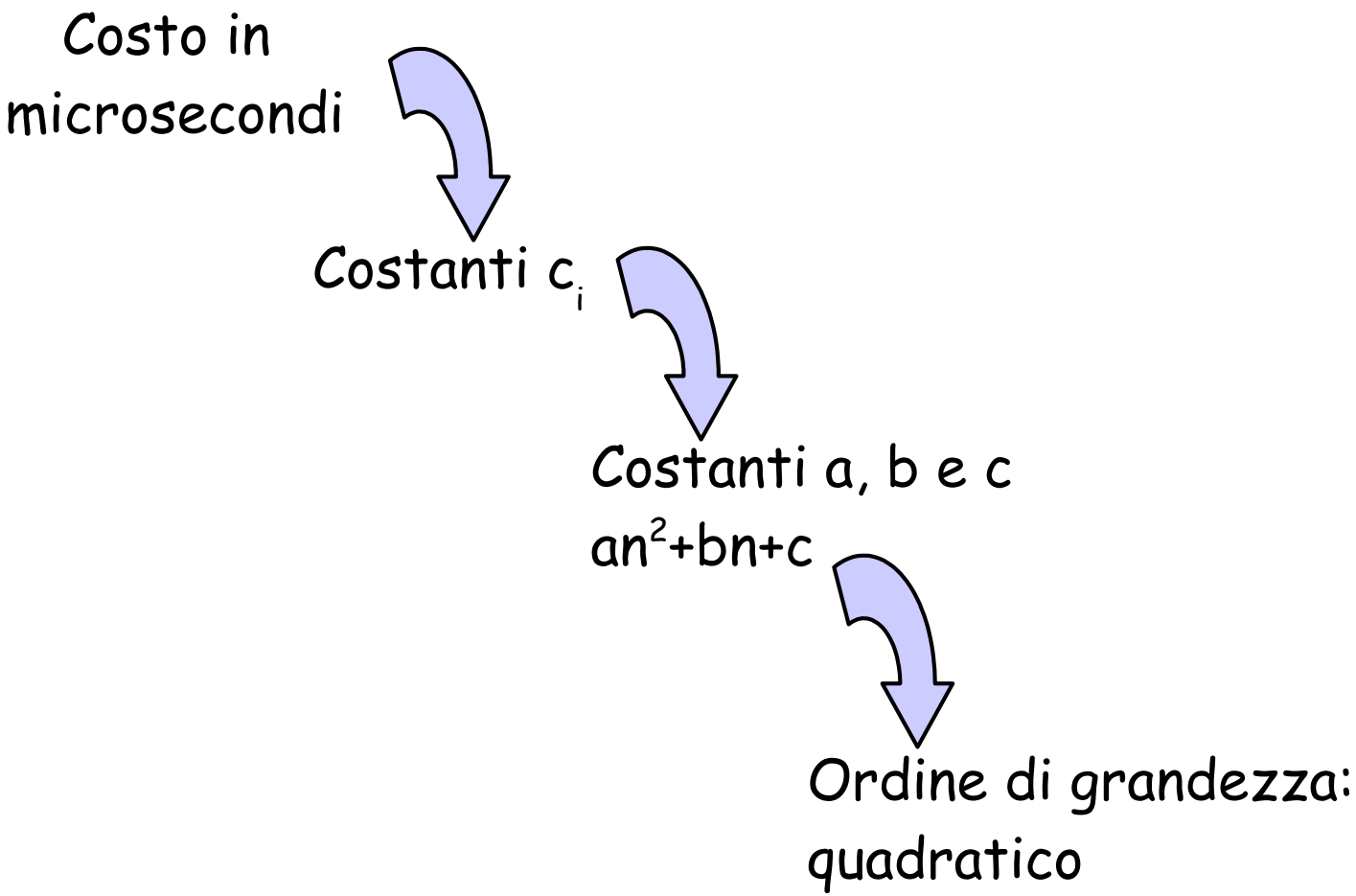
Passo 1: abbiamo ignorato il costo effettivo delle singole istruzioni introducendo delle costanti $(c_1, ..., c_8)$

Passo 2: successivamente ci siamo resi conto che anche queste costanti forniscono dettagli non necessari

- Insertion Sort: il tempo di esecuzione nel caso peggiore è $an^2 + bn + c$, dove a, b e c sono altre costanti che dipendono dai costi c_i

Passo 3: consideriamo solo l'ordine di grandezza - tasso di crescita - del tempo di esecuzione

Un graduale processo di astrazione



Analisi Asintotica

- Supponiamo di aver ricavato che il tempo (o il numero di passi) che sono necessari a completare un algoritmo sia $T(n) = 4n^2 - 2n + 2$
- Per grandi valori di n, il termine $4n^2$ diventerà preponderante rispetto agli altri, che potranno non essere considerati
  - per esempio, per n pari a 500, il termine $4n^2$ sarà pari a 1000 volte il termine $2n$
- Anche i coefficienti diventano irrilevanti se compariamo $T(n)$ con una di ordine superiore, come n^3 oppure 2^n
  - Anche se $T(n) = 1.000.000n^2$, $U(n) = n^3$ sarà maggiore di $T(n)$ per ogni n maggiore di 1.000.000 ($T(1.000.000) = 1.000.000^3 = U(1.000.000)$)

Analisi Asintotica

- Astraiamo: da alcuni dettagli irrilevanti (ex: termini di ordine diverso, costanti moltiplicative) e ci interessiamo solo al tasso di crescita - analizziamo come il tempo di esecuzione cresce asintoticamente in funzione della dimensione dell'input

- Obiettivo
  - semplificare l'analisi del tempo di esecuzione di un algoritmo prescindendo dai dettagli implementativi o di altro genere
  - classificare le funzioni in base al loro comportamento asintotico

- Asintoticamente non significa per tutti gli input. Esempio: input di piccole dimensioni

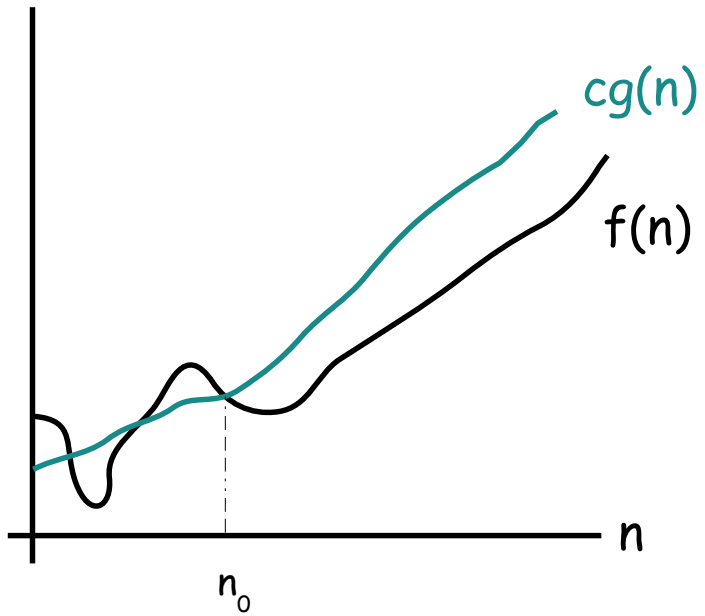
Analisi Asintotica

Importiamo alcune notazioni dalla matematica:

- O (o grande) per fornire delle delimitazioni superiori alla complessità
- Ω (omega grande) –per fornire delle delimitazione inferiori alla complessità
- Θ (theta) per fornire delle delimitazioni strette superiori ed inferiori alla complessità
- O (o grande) fornisce delle delimitazioni superiori alla complessità
- Ω (omega grande) fornisce delle delimitazione inferiori alla complessità

La notazione O grande

Siano $f(n)$ e $g(n)$ due funzioni non negative; $f(n) = O(g(n))$ se esistono due costanti **positive** c ed n_0 tali che $0 \leq f(n) \leq cg(n)$ per ogni $n \geq n_0$. Graficamente:



La notazione O grande

- Se $f(n) = O(g(n))$ allora $g(n)$ è un **limite asintotico superiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce al più come $g(n)$

La notazione O grande

- Se $f(n) = O(g(n))$ allora $g(n)$ è un **limite asintotico superiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce al più come $g(n)$
- Esempio: il limite superiore per la ricerca in un array non ordinato è $O(n)$

La notazione O grande

- Se $f(n) = O(g(n))$ allora $g(n)$ è un **limite asintotico superiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce al più come $g(n)$
- Esempio: il limite superiore per la ricerca in un array non ordinato è $O(n)$
- O viene usata nell'analisi del costo computazionale nel caso pessimo

La notazione O grande

- Se $f(n) = O(g(n))$ allora $g(n)$ è un **limite asintotico superiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce al più come $g(n)$
- Esempio: il limite superiore per la ricerca in un array non ordinato è $O(n)$
- O viene usata nell'analisi del costo computazionale nel caso pessimo
- $O(g(n))$ rappresenta l'insieme di funzioni definito come:

$$\mathrm{O(g(n))} = \{f(n) \mid \quad \text{esistono delle costanti } \textbf{positive } c \text{ ed } n_0 \text{ tali che } 0 \leq f(n) \leq cg(n) \quad \}$$

La notazione O grande

- Se $f(n) = O(g(n))$ allora $g(n)$ è un **limite asintotico superiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce al più come $g(n)$
- Esempio: il limite superiore per la ricerca in un array non ordinato è $O(n)$
- O viene usata nell'analisi del costo computazionale nel caso pessimo
- $O(g(n))$ rappresenta l'insieme di funzioni definito come:

$$\mathrm{O(g(n))} = \{f(n) \mid \quad \text{esistono delle costanti } \textbf{positive } c \text{ ed } n_0 \text{ tali che } 0 \leq f(n) \leq cg(n) \quad \}$$

- Scrivere $f(n) = O(g(n))$ è un "abuso" di notazione; avremmo dovuto scrivere $f(n) \in O(g(n))$

La notazione O grande: un esempio

Example (Provare che $f(n) = 3n^2 + 10n = O(n^2)$)

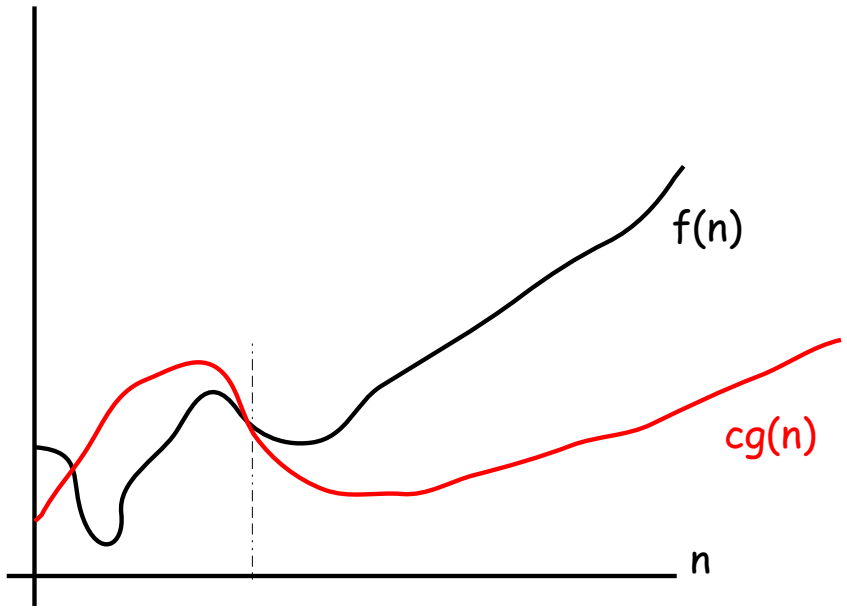
Dobbiamo dimostrare l'esistenza di costanti positive c ed n_0 tali che $0 \leq f(n) \leq cn^2$ per ogni $n \geq n_0$

Basta scegliere $c = 4$ e $n_0 = 10$; infatti:

$$
\begin{aligned}
& 3n^2 + 10n \leq cn^2 \\
& cn^2 - 3n^2 - 10n \geq 0 \\
& (c-3)n^2 - 10n \geq 0 \quad \text{scegliamo } c = 4 \\
& n^2 - 10n \geq 0 \\
& n(n-10) \geq 0 \quad \text{vera per ogni } n \geq n_0 = 10
\end{aligned}
$$

La notazione Ω grande

Siano $f(n)$ e $g(n)$ due funzioni non negative; $f(n) = \Omega(g(n))$ se esistono due costanti **positive** c ed n_0 tali che $0 \leq cg(n) \leq f(n)$ per ogni $n \geq n_0$. Graficamente:



La notazione Ω grande

- Se $f(n) = \Omega(g(n))$ allora $g(n)$ è un **limite asintotico inferiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce almeno come $g(n)$

La notazione Ω grande

- Se $f(n) = \Omega(g(n))$ allora $g(n)$ è un **limite asintotico inferiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce almeno come $g(n)$
- Esempio: il limite inferiore per la ricerca in un array non ordinato è $\Omega(n)$

La notazione Ω grande

- Se $f(n) = \Omega(g(n))$ allora $g(n)$ è un **limite asintotico inferiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce almeno come $g(n)$
- Esempio: il limite inferiore per la ricerca in un array non ordinato è $\Omega(n)$
- Ω viene usata nell'analisi del costo computazionale nel caso ottimo

La notazione Ω grande

- Se $f(n) = \Omega(g(n))$ allora $g(n)$ è un **limite asintotico inferiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce almeno come $g(n)$
- Esempio: il limite inferiore per la ricerca in un array non ordinato è $\Omega(n)$
- Ω viene usata nell'analisi del costo computazionale nel caso ottimo
- $\Omega(g(n))$ rappresenta l'insieme di funzioni definito come:

$$\Omega(g(n)) = \{f(n) \mid \text{esistono delle costanti } \textbf{positive } c \text{ ed } n_0 \text{ tali che } 0 \leq cg(n) \leq f(n) \}$$

La notazione Ω grande

- Se $f(n) = \Omega(g(n))$ allora $g(n)$ è un **limite asintotico inferiore** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce almeno come $g(n)$
- Esempio: il limite inferiore per la ricerca in un array non ordinato è $\Omega(n)$
- Ω viene usata nell'analisi del costo computazionale nel caso ottimo
- $\Omega(g(n))$ rappresenta l'insieme di funzioni definito come:

$$\Omega(g(n)) = \{f(n) \mid \text{esistono delle costanti } \textbf{positive } c \text{ ed } n_0 \text{ tali che } 0 \leq cg(n) \leq f(n) \}$$

- Scrivere $f(n) = \Omega(g(n))$ è un "abuso" di notazione; avremmo dovuto scrivere $f(n) \in \Omega(g(n))$

La notazione Ω grande: un esempio

Example (Provare che $f(n) = 3n^2 + 10n = \Omega(n^2)$)

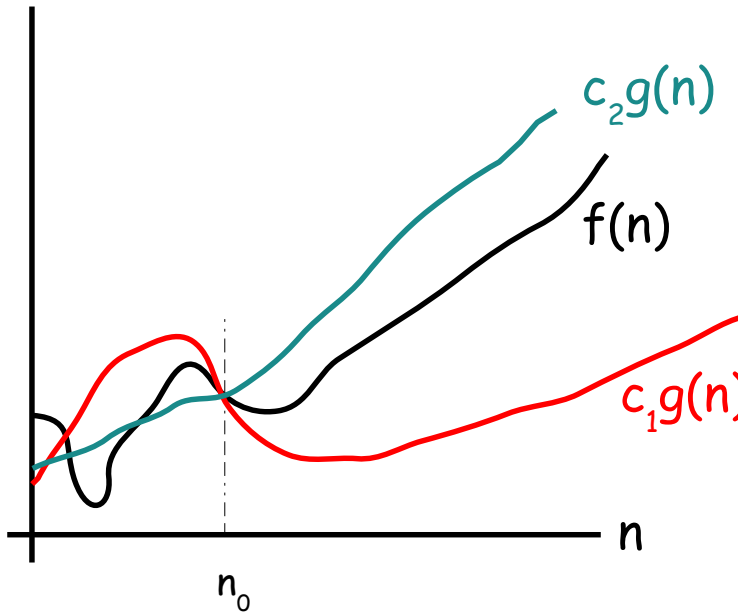
Dobbiamo dimostrare l'esistenza di costanti positive c ed n_0 tali che $0 \leq cn^2 \leq f(n)$ per ogni $n \geq n_0$

Basta scegliere $c = 3$ e $n_0 = 0$ questo perchè risulta evidente che

$$3n^2 + 10n \geq 3n^2 \text{ per ogni } n \geq n_0 = 0$$

La notazione Θ grande

Siano $f(n)$ e $g(n)$ due funzioni non negative; $f(n) = \Theta(g(n))$ se esistono tre costanti **positive** c_1, c_2 ed n_0 tali che $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ per ogni $n \geq n_0$. Graficamente:



La notazione Θ grande

- Se $f(n) = \Theta(g(n))$ allora $g(n)$ è un **limite asintotico stretto** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce esattamente come $g(n)$

La notazione Θ grande

- Se $f(n) = \Theta(g(n))$ allora $g(n)$ è un **limite asintotico stretto** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce esattamente come $g(n)$
- $\Theta(\mathrm{g(n)})$ rappresenta l'insieme di funzioni definito come:

$$\Theta(g(n)) = \{f(n) \mid \text{esistono delle costanti } \textbf{positive } c_1,\ c_2 \text{ ed } n_0 \text{ tali che } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \}$$

La notazione Θ grande

- Se $f(n) = \Theta(g(n))$ allora $g(n)$ è un **limite asintotico stretto** per $f(n)$: $f(n)$, a meno di un fattore costante, cresce esattamente come $g(n)$
- $\Theta(\mathrm{g(n)})$ rappresenta l'insieme di funzioni definito come:

$$\Theta(g(n)) = \{f(n) \mid \text{esistono delle costanti } \textbf{positive } c_1,\ c_2 \text{ ed } n_0 \text{ tali che } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \}$$

- Scrivere $f(n) = \Theta(g(n))$ è un "abuso" di notazione

Theorem

Per ogni coppia di funzioni $f(n)$ e $g(n)$, si ha che $f(n) = \Theta(g(n))$ se e soltanto se $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$

La notazione Ω grande: un esempio

Example (Provare che $f(n) = 3n^2 + 10n = \Theta(n^2)$)

Abbiamo dimostrato che $f(n) = O(n^2)$ e $f(n) = \Omega(n^2)$. Questo basta per concludere che

$$f(n) = \Theta(n^2)$$

Le notazioni "o piccolo" ed "ω piccolo"

- Siano $f(n)$ e $g(n)$ due funzioni non negative; $f(n) = o(g(n))$ se per ogni costante **positiva** c esiste una costante $n_0 > 0$ tale che $0 \leq f(n) < cg(n)$ per ogni $n \geq n_0$

- Intuitivamente, nella notazione o, la funzione $f(n)$ diventa insignificante rispetto a $g(n)$ quando n tende all'infinito – diciamo che $g(n)$ domina $f(n)$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Le notazioni "o piccolo" ed "ω piccolo"

- Siano $f(n)$ e $g(n)$ due funzioni non negative; $f(n) = \omega(g(n))$ se per ogni costante **positiva** c esiste una costante $n_0 > 0$ tale che $0 \leq cg(n) < f(n)$ per ogni $n \geq n_0$

- Intuitivamente, nella notazione ω, la funzione $f(n)$ diventa arbitrariamente grande rispetto a $g(n)$ quando n tende all'infinito – diciamo che $g(n)$ domina $f(n)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Analogie con il confronto fra numeri reali

$$
\begin{array}{lcl}
f(n) = O(g(n)) & \approx & a \leq b \\
f(n) = \Omega(g(n)) & \approx & a \geq b \\
f(n) = \Theta(g(n)) & \approx & a = b \\
f(n) = o(g(n)) & \approx & a < b \\
f(n) = \omega(g(n)) & \approx & a > b
\end{array}
$$

Transitività

$$f(n) = \Theta(g(n)) \ \wedge \ g(n) = \Theta(h(n)) \Longrightarrow f(n) = \Theta(h(n))$$

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

$$f(n) = \Omega(g(n)) \ \wedge \ g(n) = \Omega(h(n)) \Longrightarrow f(n) = \Omega(h(n))$$

$$f(n) = o(g(n)) \ \wedge \ g(n) = o(h(n)) \Longrightarrow f(n) = o(h(n))$$

$$f(n) = \omega(g(n)) \ \wedge \ g(n) = \omega(h(n)) \Longrightarrow f(n) = \omega(h(n))$$

Riflessività, Simmetria e Dualità

riflessività:

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

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

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

simmetria:

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

dualità:

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

$f(n) = o(g(n)) \iff g(n) = \omega(f(n))$