Camil Demetrescu, Irene Finocchi, Giuseppe F. Italiano

Algoritmi e Strutture Dati

Capitolo 2
Modelli di calcolo e notazione asintotica

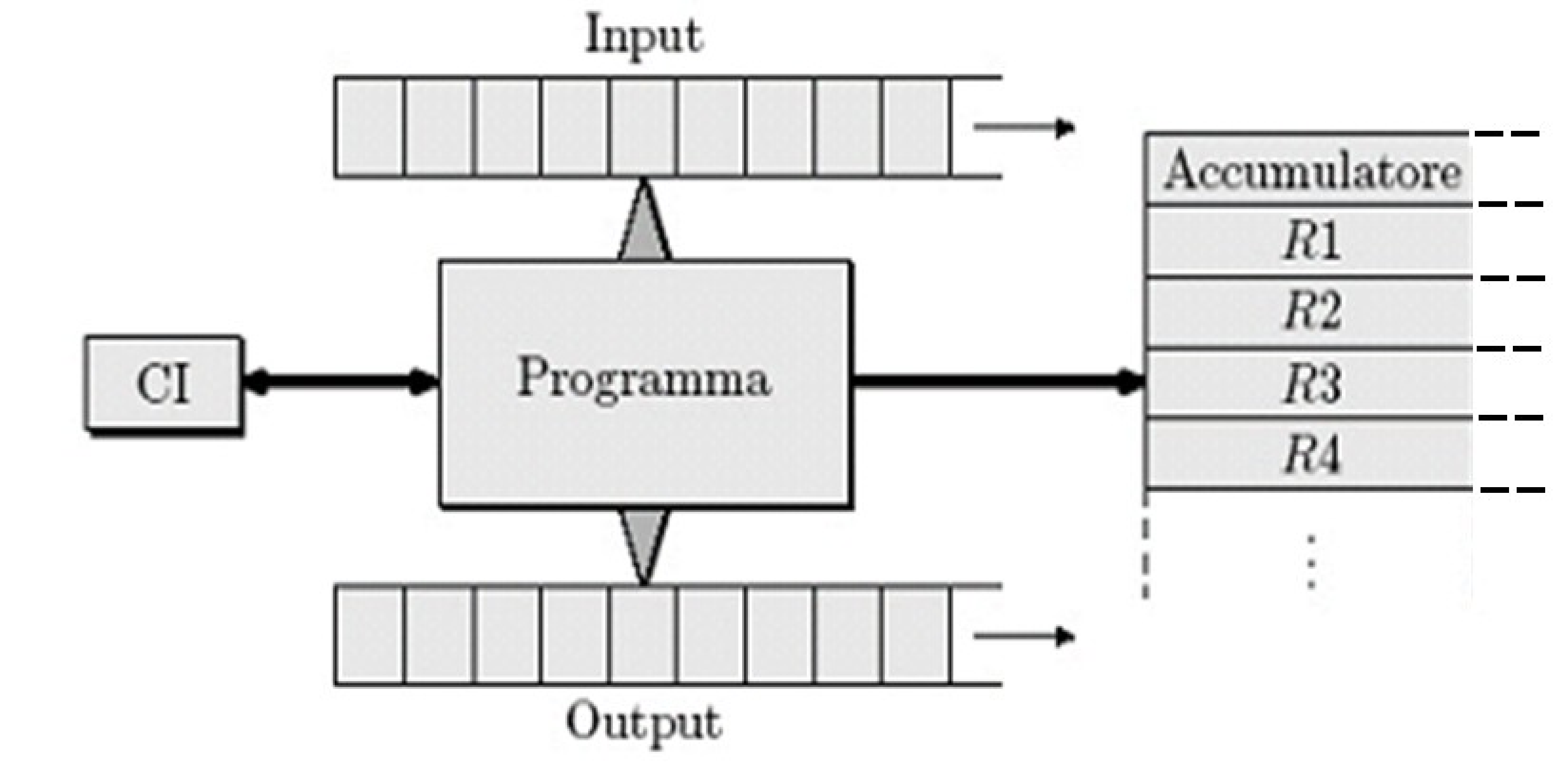
Modello di calcolo

- Per valutare la complessità temporale dei vari algoritmi `Fibonacci`, abbiamo pedissequamente contato le linee di pseudocodice di volta in volta mandate in esecuzione
- Tuttavia, contare il numero di linee dello pseudocodice di un algoritmo non è sufficiente a comprenderne la complessità temporale: quali operazioni reali si celano dietro lo pseudocodice???
- Anche l'analisi dello spazio utilizzato è stata piuttosto arbitraria: ho ipotizzato che le celle di memoria utilizzate avessero capacità infinita!
- Per valutare la complessità di un algoritmo in modo oggettivo, bisogna quindi stabilire un modello di calcolo di riferimento su cui esso viene eseguito, ovvero una macchina astratta sulla quale si definisce a priori l'insieme delle operazioni ammissibili ed eseguibili durante una computazione, specificandone i relativi costi (in termini di tempo e spazio utilizzato)
- Alcuni modelli di calcolo famosi: Macchina di Turing, Automi a Stati Finiti, Funzioni Ricorsive, Macchina a Registri, etc… (sono tutti modelli di calcolo equivalenti, cioè in grado di calcolare le stesse funzioni)

Il nostro modello di calcolo: la RAM

- La RAM (*Random Access Machine*) è un tipo particolare di macchina a registri (locazioni di memoria ad accesso diretto)
- La RAM è definita da:
  - un nastro di ingresso e uno di uscita, ciascuno strutturato in **celle**, ove saranno scritti l'input e l'output, rispettivamente; ogni cella può contenere un valore alfanumerico arbitrariamente grande
  - una memoria ad accesso diretto strutturata come un array (di dimensione infinita) di **registri**, ciascuno dei quali può contenere un valore alfanumerico arbitrariamente grande
  - un programma finito di istruzioni elementari, contenuto all'interno di una memoria addizionale (detta tavola)
  - un registro detto accumulatore (contiene gli operandi dell'istruzione corrente)
  - un registro detto contatore delle istruzioni (contiene l'indirizzo dell'istruzione corrente nella tavola)
- La RAM è un'astrazione dell'architettura di von Neumann

La RAM



Criterio di costo unitario

- Nel modello RAM a costi uniformemente unitari, le seguenti istruzioni elementari hanno un costo unitario
  - istruzione ingresso (lettura di una cella del nastro di input e suo trasferimento in memoria)
  - istruzione uscita (scrittura di una cella sul nastro di output)
  - operazione aritmetico (+,-,×,:,√, elevamento a potenza, etc.) /logica (test)
  - accesso/modifica del contenuto della memoria e dell'accumulatore
- **Complessità temporale** tempo(I) misurata come numero di istruzioni elementari eseguite sull'istanza di input I
- **Complessità spaziale** spazio(I) misurata come numero massimo di celle di memoria della RAM occupate durante l'esecuzione sull'istanza di input I

Dimensione dell'input

- Misureremo le risorse di calcolo usate da un algoritmo (tempo di esecuzione / occupazione di memoria) in funzione della dimensione dell'istanza I in input
- Esistono due modalità di caratterizzazione della dimensione dell'input:
  - Quantità di memoria effettiva utilizzata per codificare l'input; generalmente questa modalità viene adottata per problemi nei quali l'input è un unico elemento (ad esempio, nel problema di Fibonacci il valore in input è il numero n, il quale in base 2 richiede log n bit per essere rappresentato ⇒ la dimensione dell'input è proprio log n (bit))
  - Parametrizzazione della dimensione dell'input; generalmente questa modalità viene adottata per problemi nei quali l'input è un insieme di elementi (ad esempio, una sequenza di n elementi da ordinare ⇒ la dimensione dell'input diventa n)
- In tutti i problemi che incontreremo in futuro, la dimensione dell'input sarà parametrizzata attraverso il numero n di elementi dell'istanza

Notazione asintotica

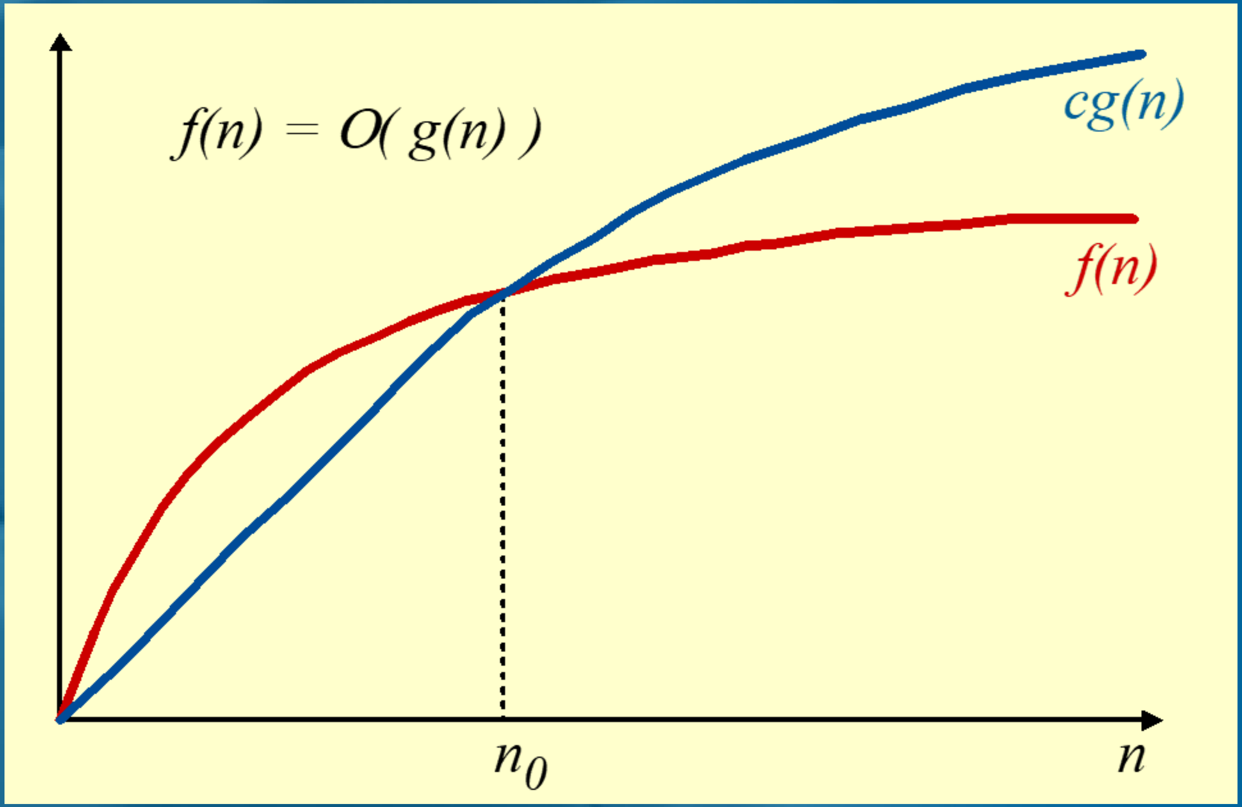
Notazione asintotica e operazioni dominanti

- Complessità temporale e spaziale saranno espresse in notazione asintotica rispetto alla dimensione dell'input
- La notazione asintotica è un'astrazione utile per descrivere l'ordine di grandezza di una funzione ignorando i dettagli non influenti, come costanti moltiplicative e termini di ordine inferiore
- **Semplificazione:** Utilizzando l'analisi asintotica, non dovremo andare a conteggiare tutte le operazioni eseguite, ma sarà sufficiente considerare le cosiddette operazioni dominanti, ovvero quelle che nel caso peggiore vengono eseguite più spesso; queste si trovano annidate nei cicli più interni dello pseudocodice che descrive l'algoritmo
- NOTA: Nel prosieguo, ci concentreremo su funzioni di variabile intera non negativa a valori reali non negativi

$$f : n \in \aleph \rightarrow \Re \geq 0$$

Notazione asintotica O ('o' grande)

$f(n) = O(g(n))$ se $\exists$ due costanti reali $c>0$ e $n_0 \geq 0$ tali che $f(n) \leq c\, g(n)$ per ogni $n \geq n_0$



Legame con il concetto di limite

- Semplificazione operativa: tutte le funzioni f, g che studieremo (polinomi, logaritmi, esponenziali) avranno un andamento 'regolare' al crescere di n, e quindi potremo stabilire che f(n) = O(g(n)) se e solo se

ovvero f(n)=O(g(n)) se e solo se f(n) è un infinito di ordine **non superiore** a g(n).

$$\lim_{n\ \infty} f(n)/g(n) = k < \infty$$

Un caso notevole: i polinomi

Sia $f(n) = a_d\, n^d + a_{d-1}\, n^{d-1} + \dots + a_0$ un polinomio di grado d (con $a_d > 0$), e sia $g(n)=n^c$. Allora, utilizzando la semplificazione di cui sopra, possiamo dimostrare che $f(n)=O(n^c)$ per ogni c⍰d, e $f(n) \neq O(n^c)$ per ogni $c<d$. Infatti:

Se c>d $\quad \lim_{n\to\infty} \dfrac{a_d n^d + \dots + a_0}{n^c} = 0 < \infty \Rightarrow f(n) = O(g(n))$

Se c=d $\quad \lim_{n\to\infty} \dfrac{a_d n^d + \dots + a_0}{n^d} = a_d < \infty \Rightarrow f(n) = O(g(n))$

Se c<d $\quad \lim_{n\to\infty} \dfrac{a_d n^d + \dots + a_0}{n^c} = \infty \Rightarrow f(n) \neq O(g(n))$

Esempio: Sia $f(n) = 2n^2 - 3n$; avremo che

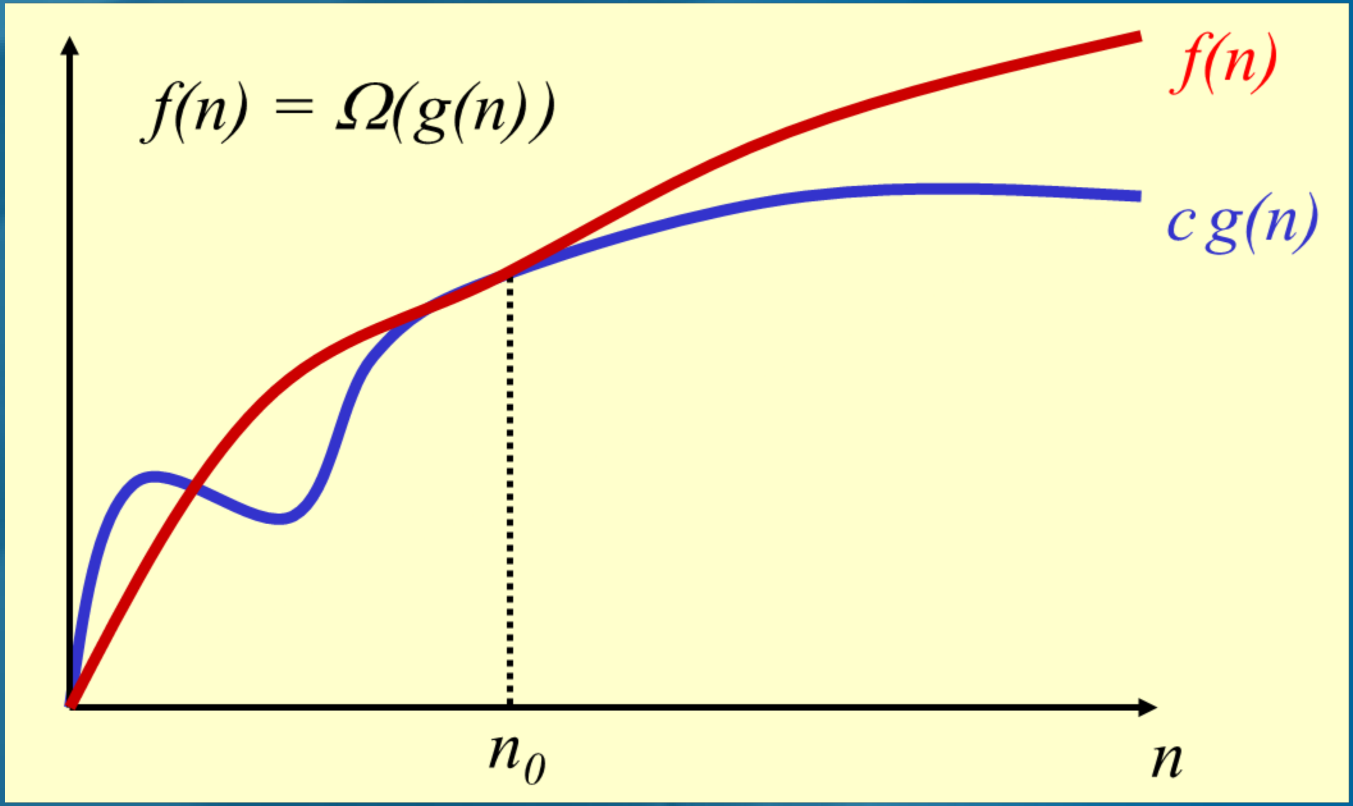
- $f(n) = O(n^3)$ (caso 1);
- $f(n) = O(n^2)$ (caso 2);
- $f(n) \neq O(n)$ (caso 3).

Osservazioni sulla notazione O

- Si noti che $O(g(n))$ è un insieme di funzioni, ovvero è (informalmente) l'insieme di funzioni che crescono al più come $g(n)$
- Ad esempio, $O(n^2)$ contiene tutti i polinomi di primo e secondo grado, nonché tutte le funzioni che crescono al più come n^2, come ad esempio $f(n) = 51$, oppure $f(n) = n \log^5 n$, oppure ancora $f(n) = n\sqrt{n}$
- Sarebbe quindi più opportuno scrivere che, ad esempio, $2n^2-3n \in O(n^2)$, ma con un piccolo abuso di notazione scriveremo sempre $2n^2-3n = O(n^2)$
- Una particolare classe asintotica è quella che denoteremo con $O(1)$: questa contiene tutte le funzioni che crescono al più come una costante, quindi ad esempio $f(n) = 51$, oppure $f(n) = 10^{11234}$, ma anche, ad esempio, $f(n) = 1/n$

Notazione asintotica Ω (omega grande)

$f(n) = \Omega(g(n))$ se $\exists$ due costanti reali $c>0$ e $n_0 \geq 0$ tali che $f(n) \geq c\, g(n)$ per ogni $n \geq n_0$



Legame con il concetto di limite

- Semplificazione operativa: tutte le funzioni che studieremo (polinomi, logaritmi, esponenziali) avranno un andamento ‘regolare’ al crescere di n, e quindi potremo stabilire che $f(n) = \Omega(g(n))$ se e solo se

ovvero $f(n)= \Omega(g(n))$ se e solo se $f(n)$ è un infinito di ordine **non inferiore** a $g(n)$.

$$\lim_{n \infty} f(n)/g(n) = k > 0$$

Un caso notevole: i polinomi

Sia $f(n) = a_d n^d + a_{d-1} n^{d-1} + \ldots + a_0$ un polinomio di grado d (con $a_d > 0$), e sia $g(n)=n^c$. Allora, utilizzando la semplificazione di cui sopra, possiamo dimostrare che $f(n)=\Omega(n^c)$ per ogni $c \le d$, e $f(n) \neq \Omega(n^c)$ per ogni $c>d$. Infatti:

$$\text{Se } c>d \quad \lim_{n\to\infty} \frac{a_d n^d + \ldots + a_0}{n^c} = 0 \Rightarrow f(n) \neq \Omega(g(n))$$

$$\text{Se } c=d \quad \lim_{n\to\infty} \frac{a_d n^d + \ldots + a_0}{n^d} = a_d > 0 \Rightarrow f(n) = \Omega(g(n))$$

$$\text{Se } c<d \quad \lim_{n\to\infty} \frac{a_d n^d + \ldots + a_0}{n^c} = \infty > 0 \Rightarrow f(n) = \Omega(g(n))$$

Esempio: Sia $f(n) = 2n^2 - 3n$; avremo che

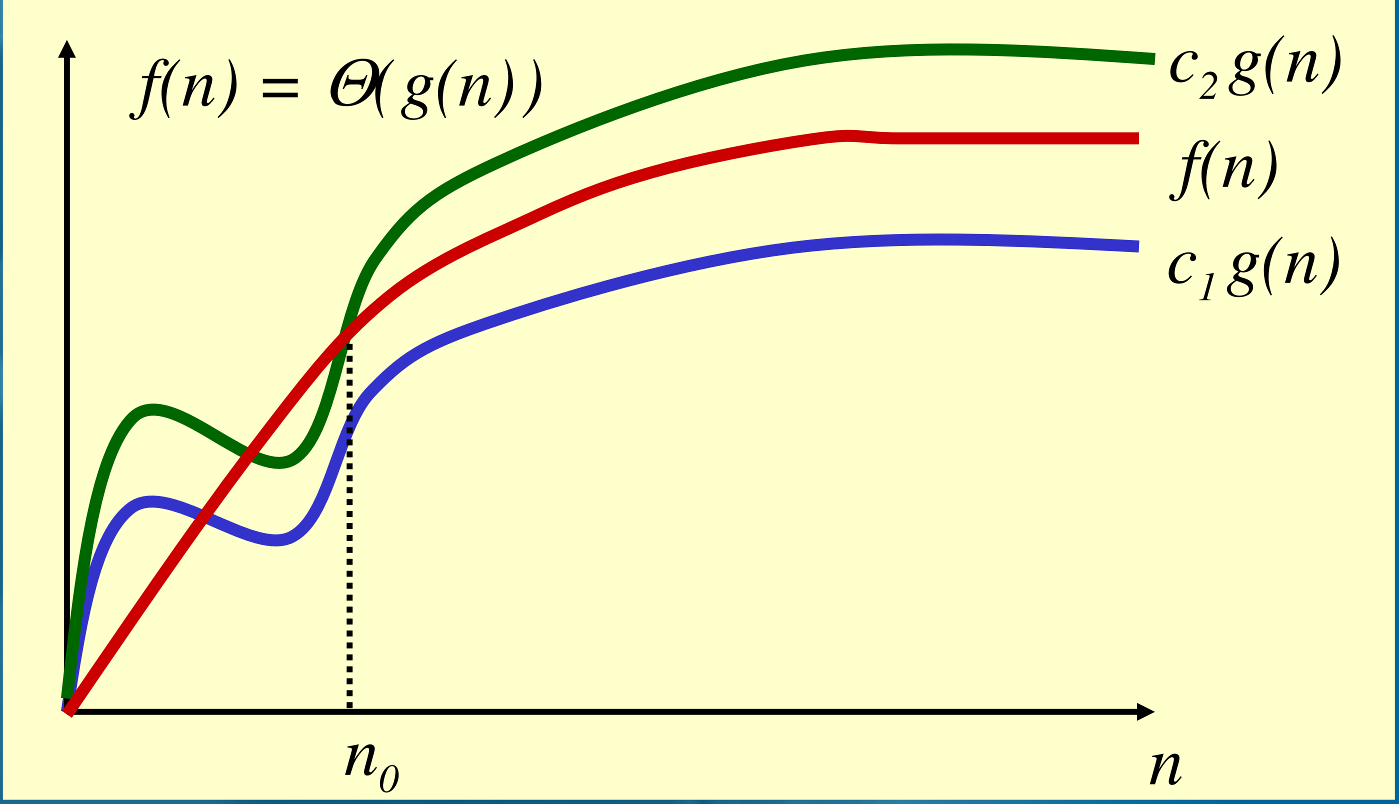
- $f(n) \neq \Omega(n^3)$ (caso 1);
- $f(n) = \Omega(n^2)$ (caso 2);
- $f(n) = \Omega(n)$ (caso 3).

Osservazioni sulla notazione Ω

- Al pari di $O(g(n))$, anche $\Omega(g(n))$ è un insieme di funzioni, ovvero è (informalmente) l'insieme di funzioni che crescono almeno come $g(n)$
- Ad esempio, $\Omega(n^2)$ contiene tutti i polinomi di grado almeno pari a 2, nonché tutte le funzioni che crescono almeno come n^2, come ad esempio $f(n) = 3n^3/\log n$, oppure $f(n) = n^2 \log^5 n$
- Sarebbe quindi più opportuno scrivere che, ad esempio, $2n^2 -5n \in \Omega(n^2)$, ma con un piccolo abuso di notazione scriveremo sempre $2n^2-5n = \Omega(n^2)$
- Una particolare classe asintotica è quella che denoteremo con $\Omega(1)$: questa contiene tutte le funzioni che crescono almeno come una costante, quindi ad esempio $f(n) = 51$, oppure $f(n) = 1/10^{11234}$, ma anche, ad esempio, $f(n) = \log n$
- Domanda: La funzione $f(n)=n^2/\log n$ è $\Omega(n^2)$?
- Domanda: La funzione costante $f(n)=0$ è $\Omega(1)$?

Notazione asintotica Θ (theta grande)

$f(n) = \Theta(g(n))$ se $\exists$ tre costanti reali $c_1, c_2 > 0$ e $n_0 \geq 0$ tali che $c_1\, g(n) \leq f(n) \leq c_2\, g(n)$ per ogni $n \geq n_0$



Legame con il concetto di limite

- Semplificazione operativa: tutte le funzioni che studieremo (polinomi, logaritmi, esponenziali) avranno un andamento ‘regolare’ al crescere di n, e quindi potremo stabilire che $f(n) = \Theta(g(n))$ se e solo se

ovvero $f(n)= \Theta(g(n))$ se e solo se $f(n)$ è un infinito dello stesso ordine di $g(n)$.

$$\lim_{n\ \infty} f(n)/g(n) = k \text{ con } 0<k<\infty$$

Un caso notevole: i polinomi

Sia $f(n) = a_d n^d + a_{d-1} n^{d-1} + \ldots + a_0$ un polinomio di grado d (con $a_d>0$), e sia $g(n)=n^c$. Allora, utilizzando la semplificazione di cui sopra, possiamo dimostrare che $f(n)=\Theta(n^c)$ solo per $c=d$, e $f(n)\neq\Theta(n^c)$ per ogni $c\neq d$. Infatti:

Se $c>d$ $\quad \lim_{n\to\infty} \frac{a_d n^d + \ldots + a_0}{n^c} = 0 \Rightarrow f(n) \neq \Theta(g(n))$

Se $c=d$ $\quad \lim_{n\to\infty} \frac{a_d n^d + \ldots + a_0}{n^d} = a_d > 0 \Rightarrow f(n) = \Theta(g(n))$

Se $c<d$ $\quad \lim_{n\to\infty} \frac{a_d n^d + \ldots + a_0}{n^c} = \infty > 0 \Rightarrow f(n) \neq \Theta(g(n))$

Esempio: Sia $f(n) = 2n^2 - 3n$; avremo che

- $f(n) \neq \Theta(n^3)$ (caso 1);
- $f(n) = \Theta(n^2)$ (caso 2);
- $f(n) \neq \Theta(n)$ (caso 3).

Osservazioni sulla notazione Θ

- Al pari di $O(g(n))$ e di $\Omega(g(n))$, anche $\Theta(g(n))$ è un insieme di funzioni, ovvero è (informalmente) l'insieme di funzioni che crescono come $g(n)$
- Ad esempio, $\Theta(n^2)$ contiene tutti i polinomi di grado pari a 2, ma anche funzioni del tipo $n^2+(\log n)/n^2$, oppure $f(n) = n^2-\sqrt{n}$
- Sarebbe quindi più opportuno scrivere che, ad esempio, $2n^2 -5n \in \Theta(n^2)$, ma con un piccolo abuso di notazione scriveremo sempre $2n^2-5n = \Theta(n^2)$
- Una particolare classe asintotica è quella che denoteremo con $\Theta(1)$: questa contiene tutte le funzioni che crescono esattamente come una costante, quindi ad esempio $f(n) = 51$, oppure $f(n) = 10^{11234}$, oppure $f(n) = 1/10^{11234}$, oppure $f(n) = 32+1/n^2$
- Domanda: La funzione $f(n)=n^2+\log n$ è $\Theta(n^2)$?
- Domanda: La funzione $f(n)=1/n$ è $\Theta(1)$?

Relazioni tra O, Ω e Θ

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

$$f(n)=O(g(n)) \nRightarrow f(n)=\Theta(g(n))$$

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

$$f(n)=\Omega(g(n)) \nRightarrow f(n)=\Theta(g(n))$$

$$f(n)=\Theta(g(n)) \Leftrightarrow f(n)=\Omega(g(n)) \text{ e } f(n)=O(g(n))$$

Camil Demetrescu, Irene Finocchi, Giuseppe F. Italiano

Notazione asintotica o ('o' piccolo)

f(n) = o(g(n)) se $\forall$ c>0, $\exists$ $n_c \geq 0$ tale che

$f(n) \leq c\, g(n)$ per ogni $n \geq n_c$

- In tutti i casi di nostro interesse, avremo che:

- Notare che

- Ad esempio, 2

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

0

$$o(g(n)) \subset O(g(n))$$

Notazione asintotica ω (omega piccolo)

$f(n) = \omega(g(n))$ se $\forall\, c>0, \exists\, n_c \geq 0$ tale che

$f(n) \geq c\, g(n)$ per ogni $n \geq n_c$

•In tutti i casi di nostro interesse, avremo che:

•Notare che

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

•Ad esempi ... $= \infty$

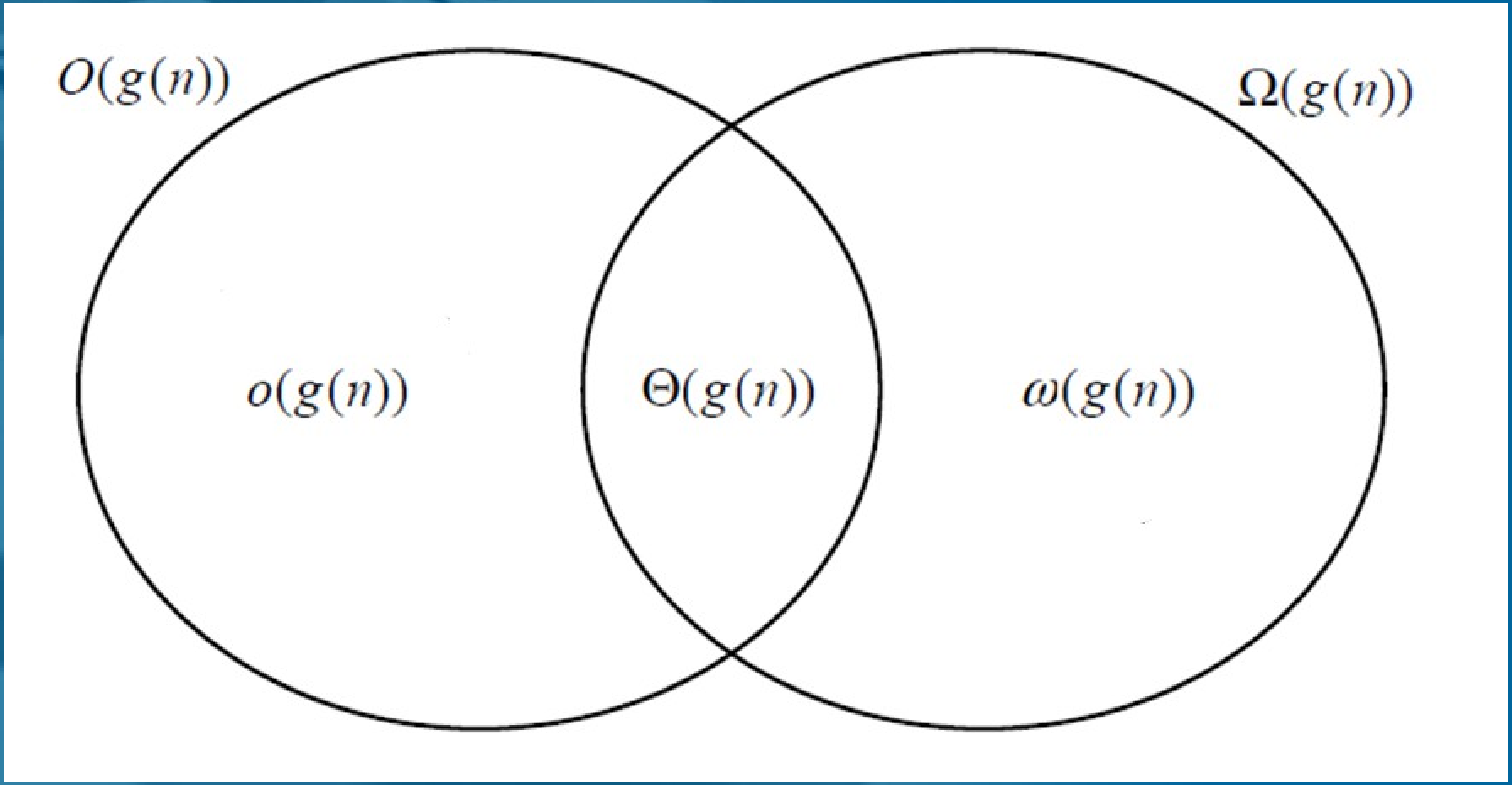
$$\omega(g(n)) \subset \Omega(g(n))$$

Analogie

$O \quad \Omega \quad \Theta \quad o \quad \omega$

$\leq \quad \geq \quad = \quad < \quad >$

Graficamente



Proprietà della notazione asintotica

Transitività

$$f(n)=\Theta(g(n)) \quad e \quad g(n)=\Theta(h(n)) \quad \Rightarrow \quad f(n)=\Theta(h(n))$$

$$f(n)=O(g(n)) \quad e \quad g(n)=O(h(n)) \quad \Rightarrow \quad f(n)=O(h(n))$$

$$f(n)=\Omega(g(n)) \quad e \quad g(n)=\Omega(h(n)) \quad \Rightarrow \quad f(n)=\Omega(h(n))$$

$$f(n)=o(g(n)) \quad e \quad g(n)=o(h(n)) \quad \Rightarrow \quad f(n)=o(h(n))$$

$$f(n)=\omega(g(n)) \quad e \quad g(n)=\omega(h(n)) \quad \Rightarrow \quad f(n)=\omega(h(n))$$

Riflessività

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

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

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

Simmetria

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

Simmetria trasposta

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

$$f(n)=o(g(n)) \quad \Leftrightarrow \quad g(n)=\omega(f(n))$$

Relazioni asintotiche notevoli

Polinomi

$\mathcal{P}(n) = a_d\, n^d + a_{d-1}\, n^{d-1} + \ldots + a_0$

$a_d > 0$

$\mathcal{P}(n) = O(n^d)$, $\mathcal{P}(n) = \Omega(n^d) \Rightarrow \mathcal{P}(n) = \Theta(n^d)$

$\mathcal{P}(n) = O(n^c)\ \forall\ c \geq d$, $\mathcal{P}(n) \neq O(n^c)\ \forall\ c<d$

$\mathcal{P}(n) = \Omega(n^c)\ \forall\ c \leq d$, $\mathcal{P}(n) \neq \Omega(n^c)\ \forall\ c>d$

$\mathcal{P}(n) = o(n^c)\ \forall\ c>d$, $\mathcal{P}(n) = \omega(n^c)\ \forall\ c<d$

Esponenziali

$f(n) = a^n$

$a > 1$

$$\lim_{n\to\infty} \frac{a^n}{n^d} = \infty\ \forall\ d \geq 0$$

$a^n = \omega(n^d)\ \forall\ d \geq 0$

$\Rightarrow a^n = \Omega(n^d)\ \forall\ d \geq 0$

Logaritmi

$f(n) = \log_b n \quad b>1$

$$\lim_{n\to\infty} \frac{(\log_b n)^c}{n^d} = 0,\ \forall c,\ d \geq 1$$

$(\log_b n)^c = o(n^d)\ \forall\ c,d \geq 1$

$\Rightarrow (\log_b n)^c = O(n^d)\ \forall\ c,d \geq 1$

Fattoriali

$f(n) = n! = n\cdot(n-1)\cdot(n-2)\cdot\ldots\ldots\cdot 2\cdot 1$

$n! = o(n^n) \Rightarrow n! = O(n^n)$

Più precisamente: $n! = \Theta(n^{n+½}\, e^{-n})$

$n! = \omega(a^n) \Rightarrow n! = \Omega(a^n)\ \forall\ a>0$

Approfondimento: notazione asintotica per gli algoritmi di Fibonacci (in funzione del **valore** di input)

	Numero di linee di codice	Occupazione di memoria
`fibonacci1`	$1 = \Theta(1)$	$1 = \Theta(1)$
`fibonacci2`	$3F_n\text{-}2 \approx \quad {}^n = \Theta(\ {}^n)$	$\approx \Phi^n = \Theta(\Phi^n)$ (*)
`fibonacci3`	$2n = \Theta(n)$	$n+1 = \Theta(n)$
`fibonacci4`	$4n\text{-}5 = \Theta(n)$	$4 = \Theta(1)$
`fibonacci5`	$2n+1 = \Theta(n)$	$5 = \Theta(1)$
`fibonacci6`	$\log n \leq T(n) \leq 5(1+\log n)$ $\Rightarrow T(n)=\Theta(\log n)$	$\approx \log n = \Theta(\log n)$ (*)

* per le variabili di lavoro delle chiamate ricorsive

Domande di approfondimento

- Perché la tabella precedente non illustra correttamente la complessità temporale dei vari algoritmi di Fibonacci?
- Qual è la complessità temporale in notazione asintotica degli algoritmi `Fibonacci2, Fibonacci4 e Fibonacci6` in funzione della dimensione dell'input e non del valore di input?