

[FIM] FONDAMENTI DI INFORMATICA per medicina e chirurgia high tech

L04: Complexity

Dott. Giorgio De Magistris

demagistris@diag.uniroma1.it

Prof. Christian Napoli

c.napoli@uniroma1.it

CORSO DI LAUREA IN MEDICINA E CHIRURGIA HIGH TECH



SAPIENZA
UNIVERSITÀ DI ROMA

I3S

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE, INFORMATICA E STATISTICA

DIAG

DIPARTIMENTO DI INGEGNERIA INFORMATICA, AUTOMATICA E GESTIONALE

TUTTI I DIRITTI RELATIVI AL PRESENTE MATERIALE DIDATTICO ED AL SUO CONTENUTO SONO RISERVATI A SAPIENZA E AI SUOI AUTORI (O DOCENTI CHE LO HANNO PRODOTTO). È CONSENTITO L'USO PERSONALE DELLO STESSO DA PARTE DELLO STUDENTE A FINI DI STUDIO. NE È VIETATA NEL MODO PIÙ ASSOLUTO LA DIFFUSIONE, DUPLICAZIONE, CESSIONE, TRASMISSIONE, DISTRIBUZIONE A TERZI O AL PUBBLICO PENA LE SANZIONI APPLICABILI PER LEGGE

Analyzing algorithms

- If I want to compare two algorithms that solve the same problem
- How can I decide which algorithm is the best?
- An algorithm is evaluated based on the resources it occupies, usually time and space



Cost model

- Assume that common operations implemented in RAM have a constant time
- Each instruction is executed in a constant time
- The running time of the algorithm is the time of each instruction multiplied by the number of time each instruction is executed
- The running time is expressed as a function of the size of the input
- In many cases the size of the input is the number of elements given in input to the algorithm

Example Insertion Sort

INSERTION-SORT(<i>A</i>)	<i>cost</i>	<i>times</i>
1 for <i>j</i> = 2 to <i>A.length</i>	c_1	n
2 $key = A[j]$	c_2	$n - 1$
3 // Insert $A[j]$ into the sorted sequence $A[1 \dots j - 1]$.	0	$n - 1$
4 $i = j - 1$	c_4	$n - 1$
5 while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^n t_j$
6 $A[i + 1] = A[i]$	c_6	$\sum_{j=2}^n (t_j - 1)$
7 $i = i - 1$	c_7	$\sum_{j=2}^n (t_j - 1)$
8 $A[i + 1] = key$	c_8	$n - 1$

$$\begin{aligned} T(n) = & c_1 n + c_2(n - 1) + c_4(n - 1) + c_5 \sum_{j=2}^n t_j + c_6 \sum_{j=2}^n (t_j - 1) \\ & + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n - 1) . \end{aligned}$$

Example Insertion Sort

- If we consider the following identities

$$\sum_{j=2}^n j = \frac{n(n+1)}{2} - 1$$

and

$$\sum_{j=2}^n (j-1) = \frac{n(n-1)}{2}$$

- We obtain

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \left(\frac{n(n+1)}{2} - 1 \right) \\ &\quad + c_6 \left(\frac{n(n-1)}{2} \right) + c_7 \left(\frac{n(n-1)}{2} \right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n \\ &\quad - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

Order of growth

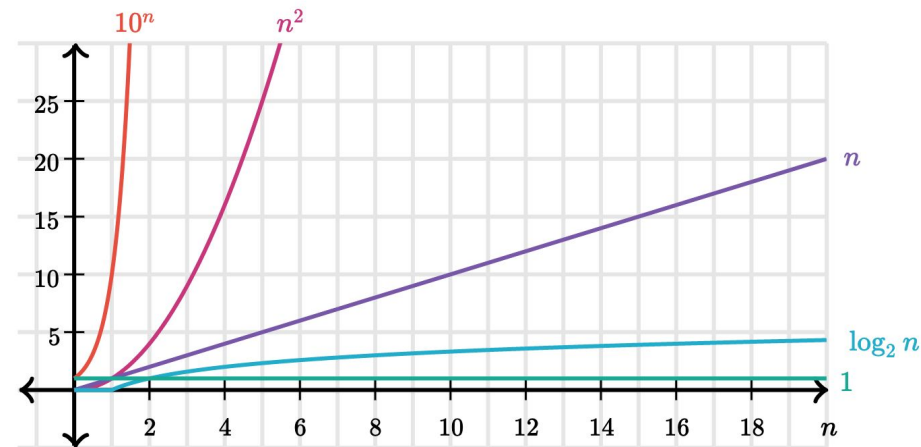
- Assuming that the computer performs one operation in 0.1 ms (10000 operations per second)

	<i>n</i>			
	10	50	100	1,000
lg <i>n</i>	0.0003 sec	0.0006 sec	0.0007 sec	0.0010 sec
<i>n</i>^½	0.0003 sec	0.0007 sec	0.0010 sec	0.0032 sec
<i>n</i>	0.0010 sec	0.0050 sec	0.0100 sec	0.1000 sec
<i>n</i> lg <i>n</i>	0.0033 sec	0.0282 sec	0.0664 sec	0.9966 sec
<i>n</i>²	0.0100 sec	0.2500 sec	1.0000 sec	100.00 sec
<i>n</i>³	0.1000 sec	12.500 sec	100.00 sec	1.1574 day
<i>n</i>⁴	1.0000 sec	10.427 min	2.7778 hrs	3.1710 yrs
<i>n</i>⁶	1.6667 min	18.102 day	3.1710 yrs	3171.0 cen
2^{<i>n</i>}	0.1024 sec	35.702 cen	4×10 ¹⁶ cen	1×10 ¹⁶⁶ cen
<i>n</i>!	362.88 sec	1×10 ⁵¹ cen	3×10 ¹⁴⁴ cen	1×10 ²⁵⁵⁴ cen

Image credit: <http://www.ccs.neu.edu/home/jaa/CS7800.12F/Information/Handouts/order.html>

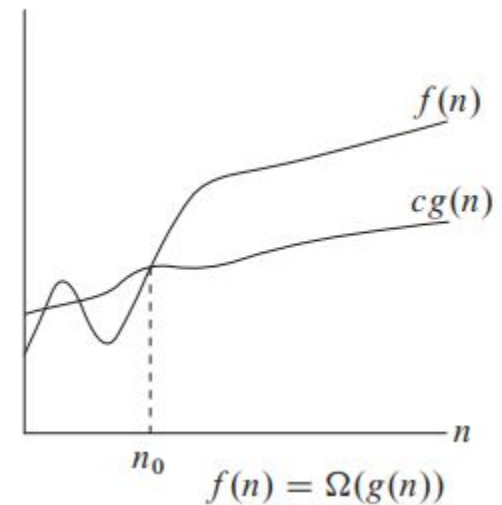
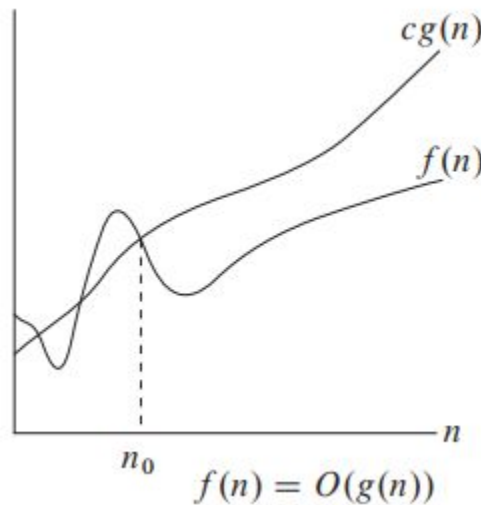
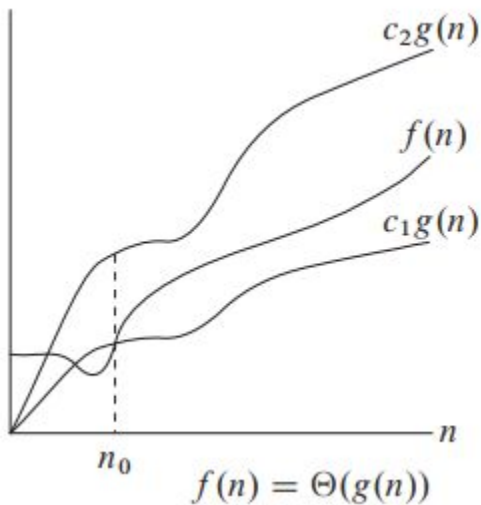
Asymptotic Notation

- When the input is large, the constants do not affect much the value of the running time
- When analyzing the algorithm for large input the asymptotic notation is used:
 - we consider only the order of growth of the running time
 - i.e. the leading term of the formula
- Different asymptotic notations exist: Θ -notation, O -notation, Ω -notation



Asymptotic Notation

- A function $f(n)$ belongs to $\Theta(g(n))$ if there are two constants such that for large n we have that $c_1g(n) \leq f(n) \leq c_2g(n)$
- A function $f(n)$ belongs to $O(g(n))$ if it is upper bounded by $g(n)$, i.e. for large n we have that $0 \leq f(n) \leq cg(n)$
- A function $f(n)$ belongs to $\Omega(g(n))$ if it is lower bounded by $g(n)$, i.e. for large n we have that $0 \leq cg(n) \leq f(n)$



References

- Cormen, Thomas H., et al. Introduction to algorithms. MIT press, 2022.

Slides distribuite con Licenza Creative Commons (CC BY-NC-ND 4.0) Attribuzione - Non commerciale - Non opere derivate 4.0 Internazionale

PUOI CONDIVIDERLE ALLE SEGUENTI CONDIZIONI

(riprodurre, distribuire, comunicare o esporre in pubblico, rappresentare, eseguire e recitare questo materiale con qualsiasi mezzo e formato)

Attribuzione*

Devi riconoscere una menzione di paternità adeguata, fornire un link alla licenza e indicare se sono state effettuate delle modifiche. Puoi fare ciò in qualsiasi maniera ragionevole possibile, ma non con modalità tali da suggerire che il licenziante avalli te o il tuo utilizzo del materiale.

Non Commerciale

Non puoi utilizzare il materiale per scopi commerciali.

Non opere derivate

Se remixi, trasformi il materiale o ti basi su di esso, non puoi distribuire il materiale così modificato.

Divieto di restrizioni aggiuntive

Non puoi applicare termini legali o misure tecnologiche che impongano ad altri soggetti dei vincoli giuridici a questa licenza