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

L04: Complexity

Dott. Giorgio De Magistris

demagistris@diag.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

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(A)	*cost*	*times*
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
$i = j - 1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	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)

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

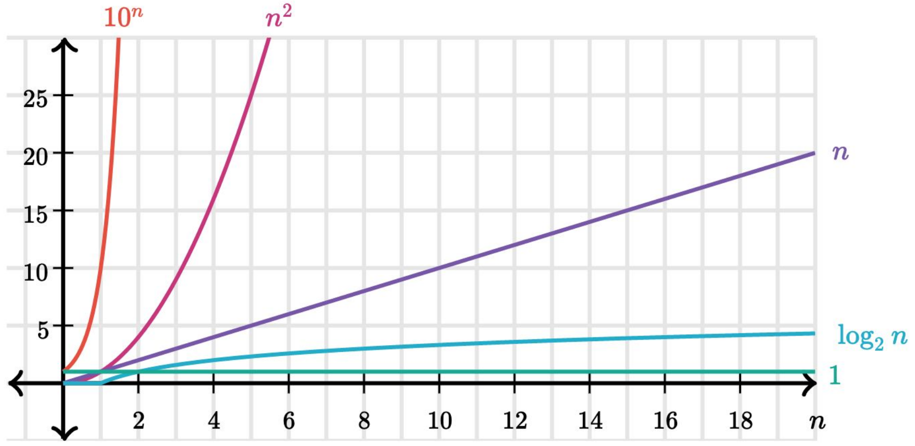
- When the input is large, the constants do not affect much the value of the running time

- 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

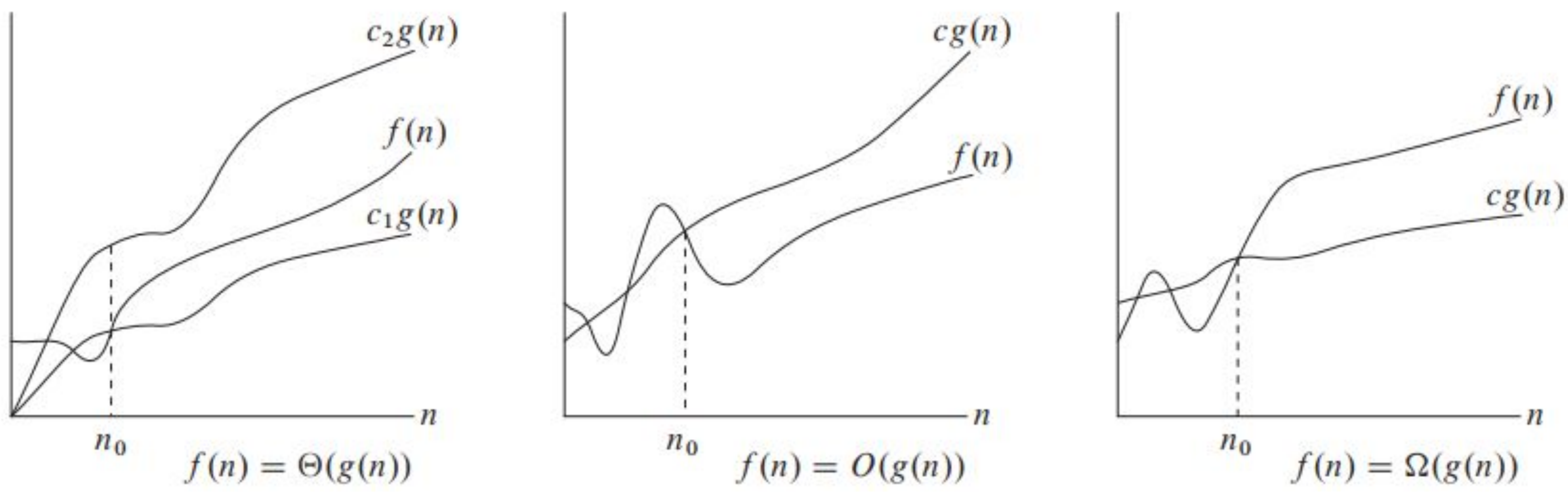


Asymptotic Notation

- A function f(n) belongs to Θ(g(n)) if there are two constants such that for large n we have that c1g(n) <= f(n) <= c2g(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 <= f(n) <= cg(n)

- A function f(n) belongs to Ω(g(n)) if it is lower bounded by g(n), i.e. for large n we have that 0<= cg(n) <= f(n)



References

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