ESTRUTURAS DE DADOS II

MSC. DANIELE CARVALHO OLIVEIRA

DOUTORANDA EM CIÊNCIA DA COMPUTAÇÃO - USP

MESTRE EM CIÊNCIA DA COMPUTAÇÃO – UFU

BACHAREL EM CIÊNCIA DA COMPUTAÇÃO - UFJF

PRINCÍPIO DE ANÁLISE DE ALGORITMOS

3 NOTAÇÃO O – BIG-OH

- **Notação O** é utilizada para ter uma estimativa superior do tempo T(n) de execução, em termos de funções do tipo n^k, log n, 2^n, cujas tendências de crescimento seguem padrões distintos.

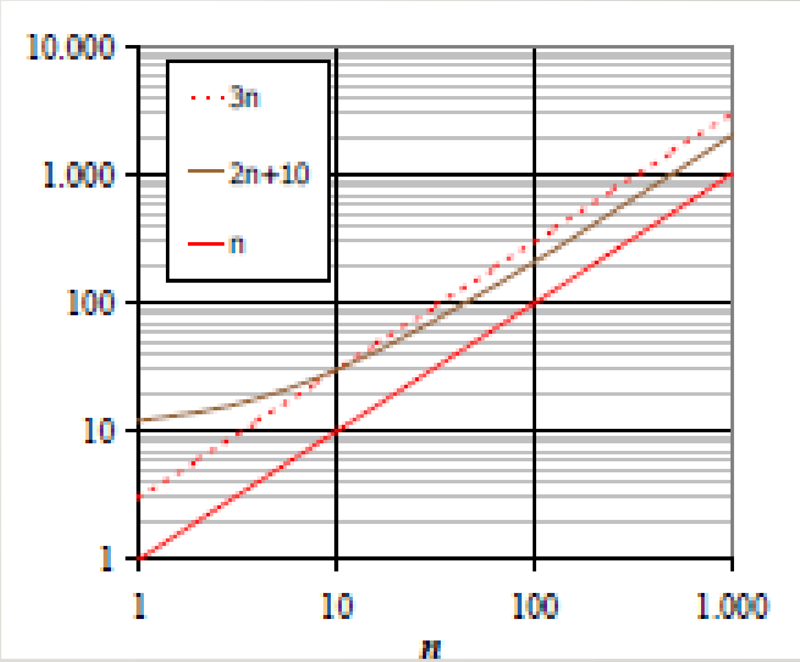
- Insertion Sort - No melhor caso:
  - T(n) = O(n)

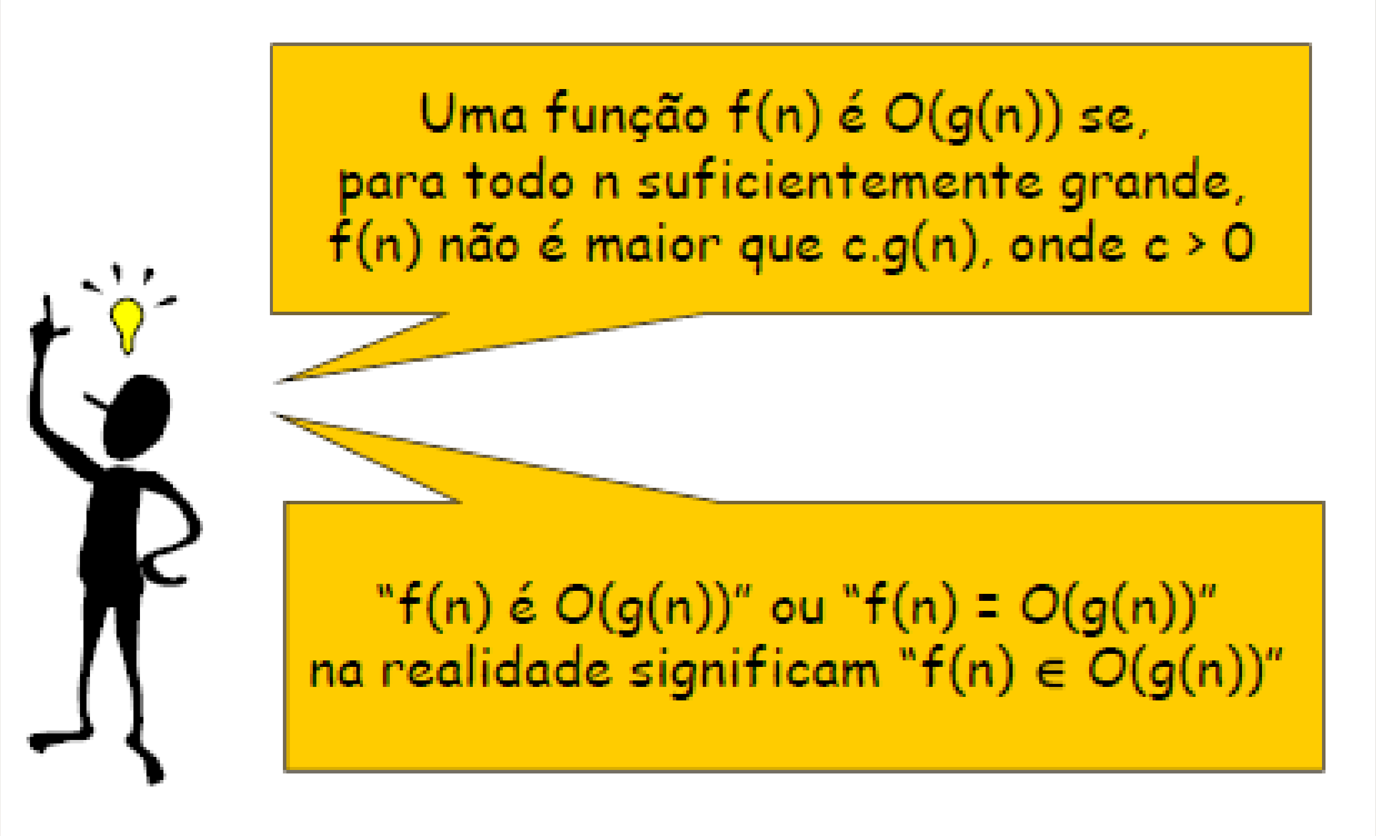
- Insertion Sort - No pior caso:
  - $T(n) = O(n^2)$

4 NOTAÇÃO O

- Dadas as funções $f(n)$ e $g(n)$, dizemos que $f(n)$ é $O(g(n))$ se existem duas constantes positivas c e n_0 tais que:
  $f(n) \leq c.g(n), \forall n \geq n_0$

- Exemplo: $2n + 10$ é $O(n)$
  - $2n + 10 \leq c.n$
  - $(c - 2).n \geq 10$
  - $n \geq 10/(c - 2)$
  - Uma possível escolha: $c = 3$ e $n_0 = 10$
  - Basta que essas constantes existam!!




Uma função f(n) é O(g(n)) se, para todo n suficientemente grande, f(n) não é maior que c.g(n), onde c > 0
"f(n) é O(g(n))" ou "f(n) = O(g(n))" na realidade significam "f(n) ∈ O(g(n))"

ATENÇÃO

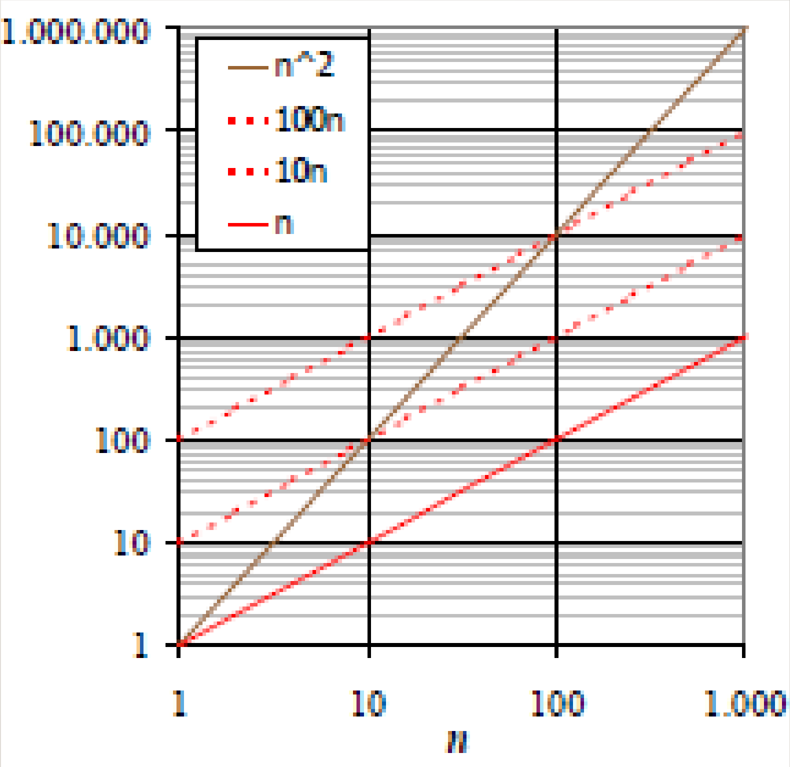
- No uso da notação O, o sinal de igualdade não tem o seu significado habitual

$$\boxed{10\,n^2 + 10 \log n = O(n^2)} \qquad \boxed{2\,n^2 - 3 = O(n^2)}$$

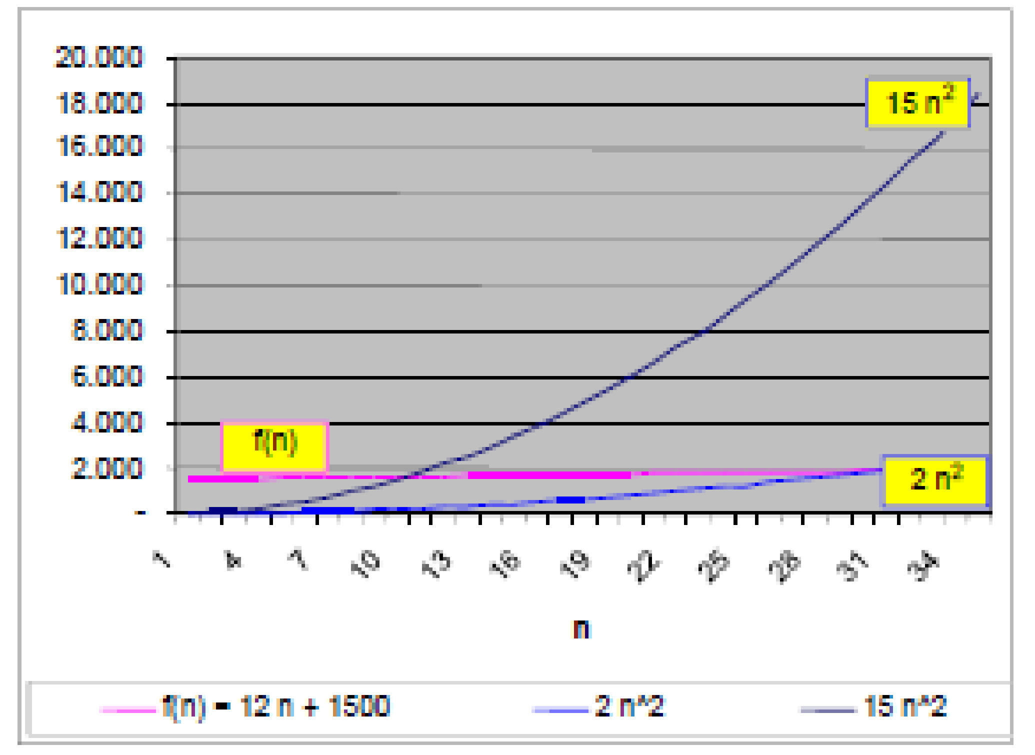
$$\cancel{\boxed{10\,n^2 + 10 \log n = 2\,n^2 - 3}}$$

EXEMPLO

- n^2 não é O(n)
  - $n^2 \leq c.n$
  - $n \leq c$
- A inequação acima não pode ser sempre satisfeita, pois c é uma constante e n não
- Qualquer c escolhido poderia ser superado por n: basta escolher $n_0 = c + 1$



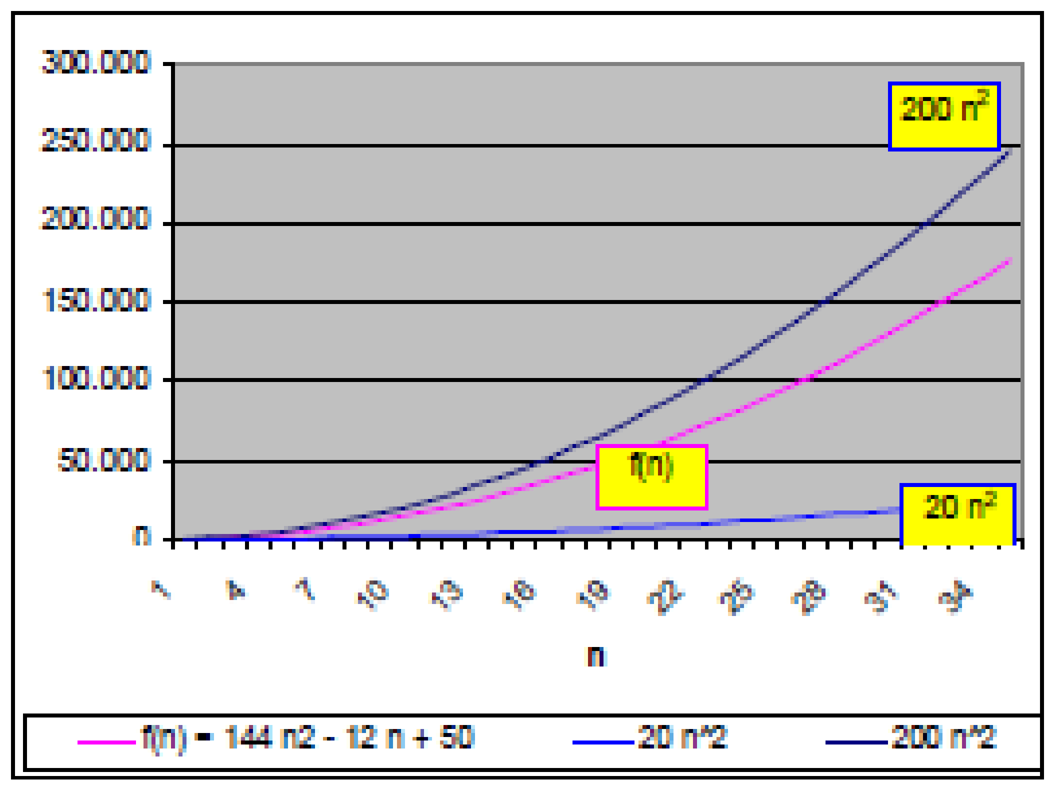
$f(n) = 12n + 1500$ É $O(n^2)$?



Sim!

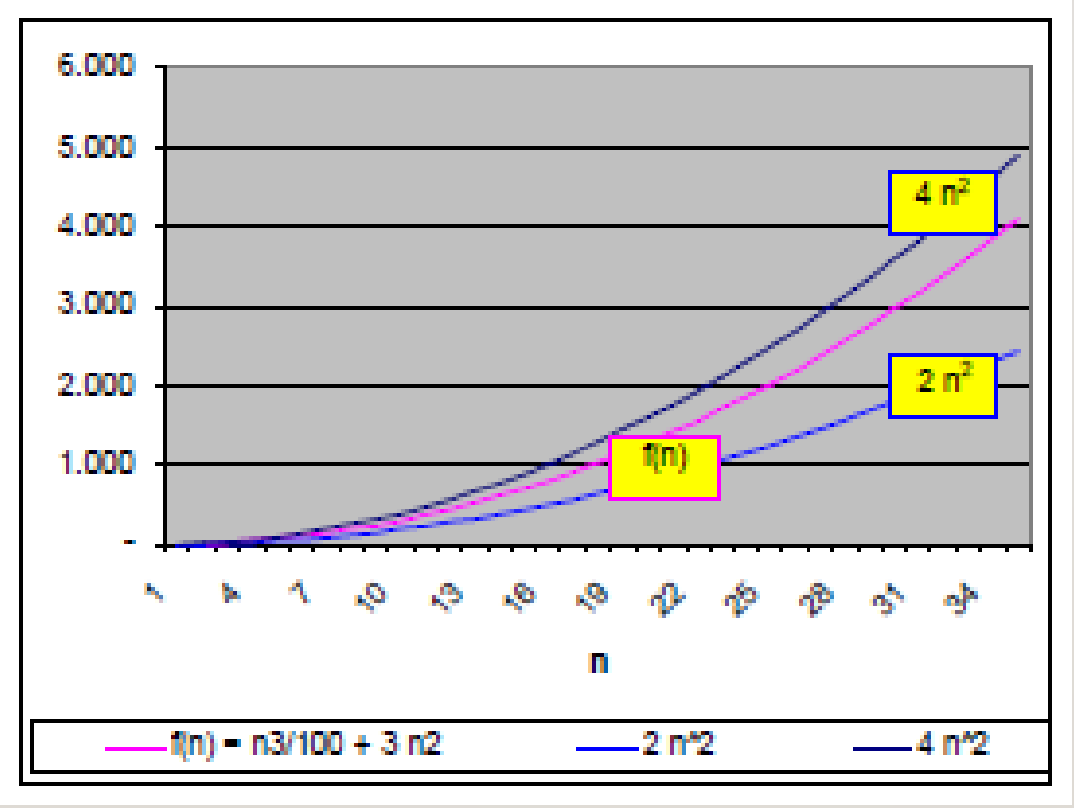
9

$f(n) = 144\,n^2 - 12\,n + 50$ É $O(n^2)$?



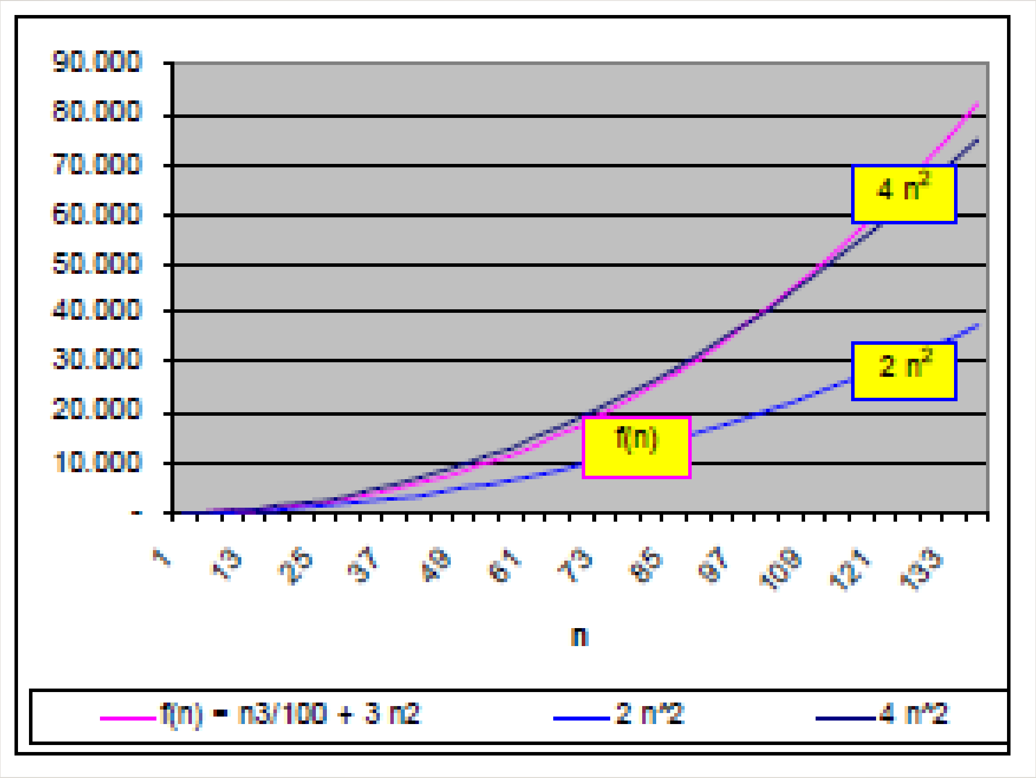
Sim!

$f(n) = n^3/100 + 3\ n^2$ É $O(n^2)$?



Sim?

Será???


90.000
80.000
70.000
60.000
50.000
40.000
30.000
20.000
10.000
-
1
13
25
37
49
61
73
85
97
109
121
133
n
4 n^2
2 n^2
f(n)
f(n) = n3/100 + 3 n2
2 n^2
4 n^2


Não!!!

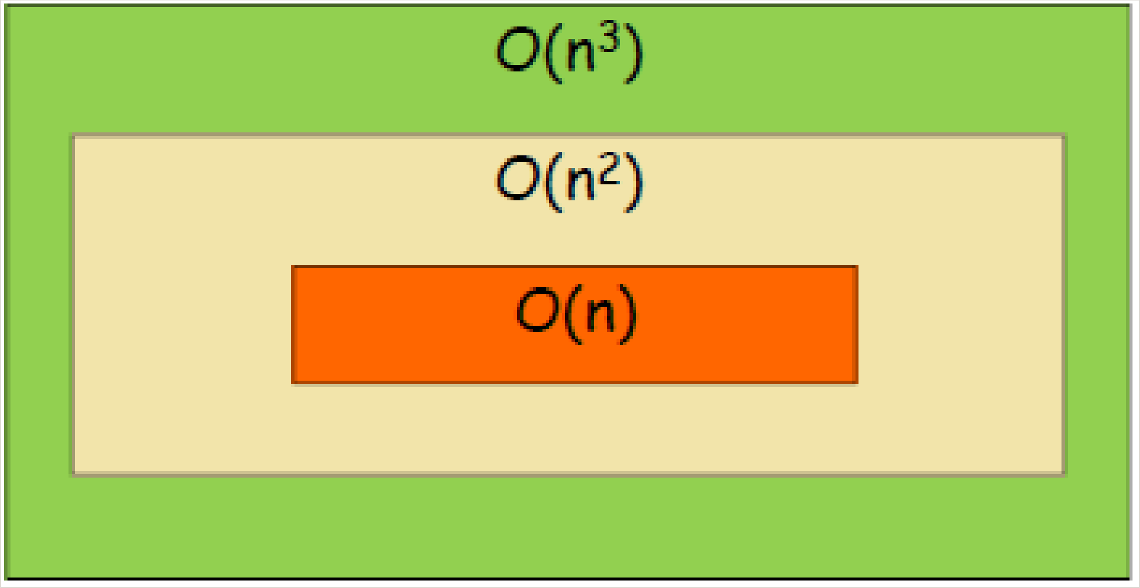
A NOTAÇÃO O E A TAXA DE CRESCIMENTO

- A notação O fornece um limite superior para a taxa de crescimento de uma determinada função.
- A afirmação "f(n) é O(g(n))" significa que a taxa de crescimento de f(n) não é maior que a de g(n).
- A notação O é utilizada para ordenar as funções de acordo com as suas correspondentes taxas de crescimento.

	f(n) é O(g(n))?	g(n) é O(f(n))?
Se g(n) cresce mais que f(n):	Sim	Não
Se f(n) cresce mais que g(n):	Não	Sim
Se f(n) e g(n) têm a mesma taxa:	Sim	Sim

13 HIERARQUIA DE FUNÇÕES

Com relação às funções polinomiais, é fácil concluir que $O(n) \subset O(n^2) \subset O(n^3) \subset O(n^4) \subset O(n^5) \subset \ldots$



14 HIERARQUIA DE FUNÇÕES

- Portanto, toda função $O(n^i)$ é também $O(n^{i+1})$, onde i ≥ 0
- No uso da notação O, consideramos apenas valores grandes de n, ou seja, $n \to \infty$

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

15 ALGUMAS REGRAS DA NOTAÇÃO O

- Se $f(n)$ é uma função polinomial de grau d, então $f(n)$ é $O(n^d)$:
  - pode-se descartar os termos de menor ordem;
  - pode-se descartar os termos constantes.

- Convém utilizar a menor ordem:
- “$2n$ é $O(n)$” é preferível a “$2n$ é $O(n^2)$”
- “$3n + 5$ é $O(n)$” é preferível a “$3n + 5$ é $O(3n)$”

EXEMPLOS

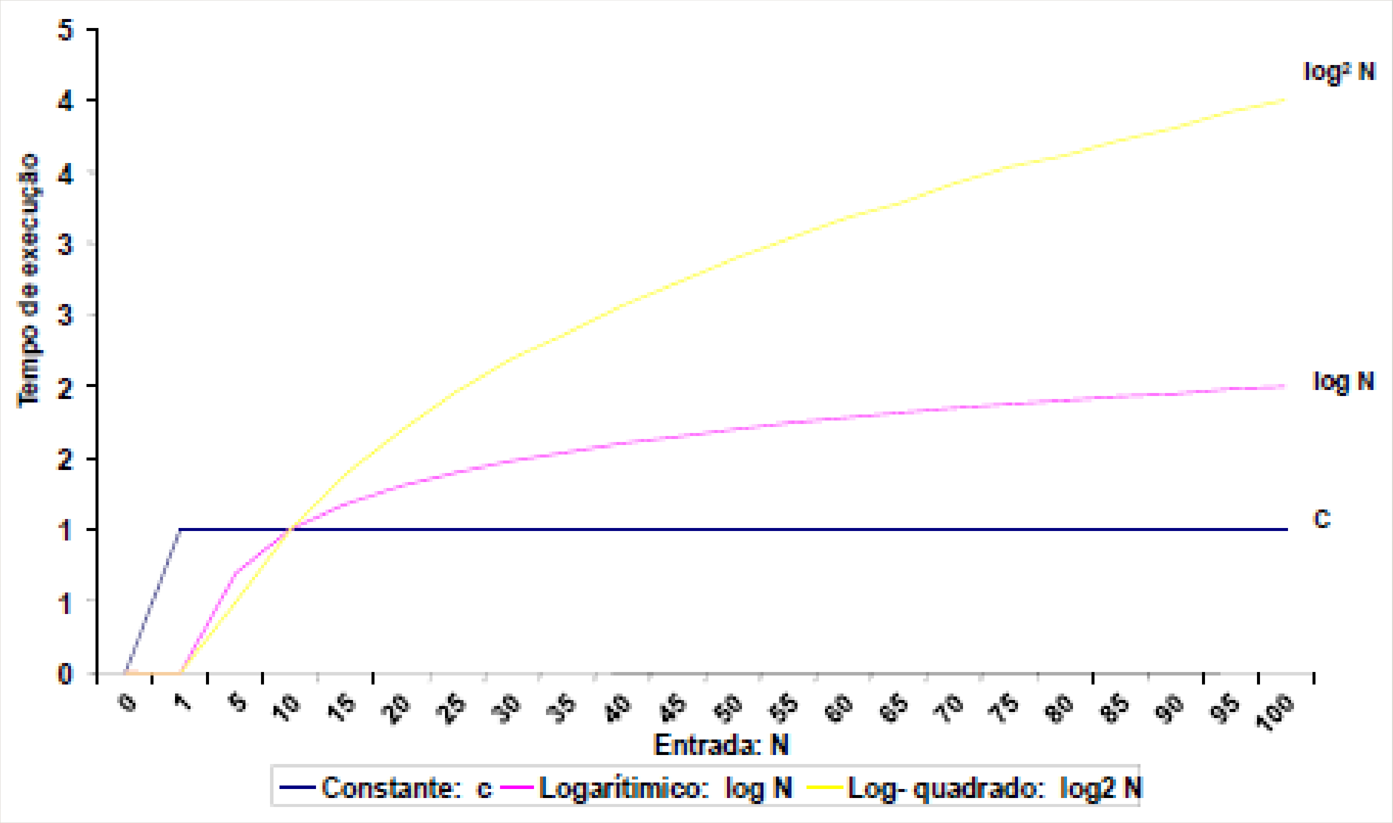
$6n^4 + 12n^3 + 12$ $\in O(n^4)$ $\in O(n^5)$ $\notin O(n^3)$

$3n^2 + 12n.\log n$ $\in O(n^2)$ $\in O(n^4)$ $\notin O(n.\log n)$

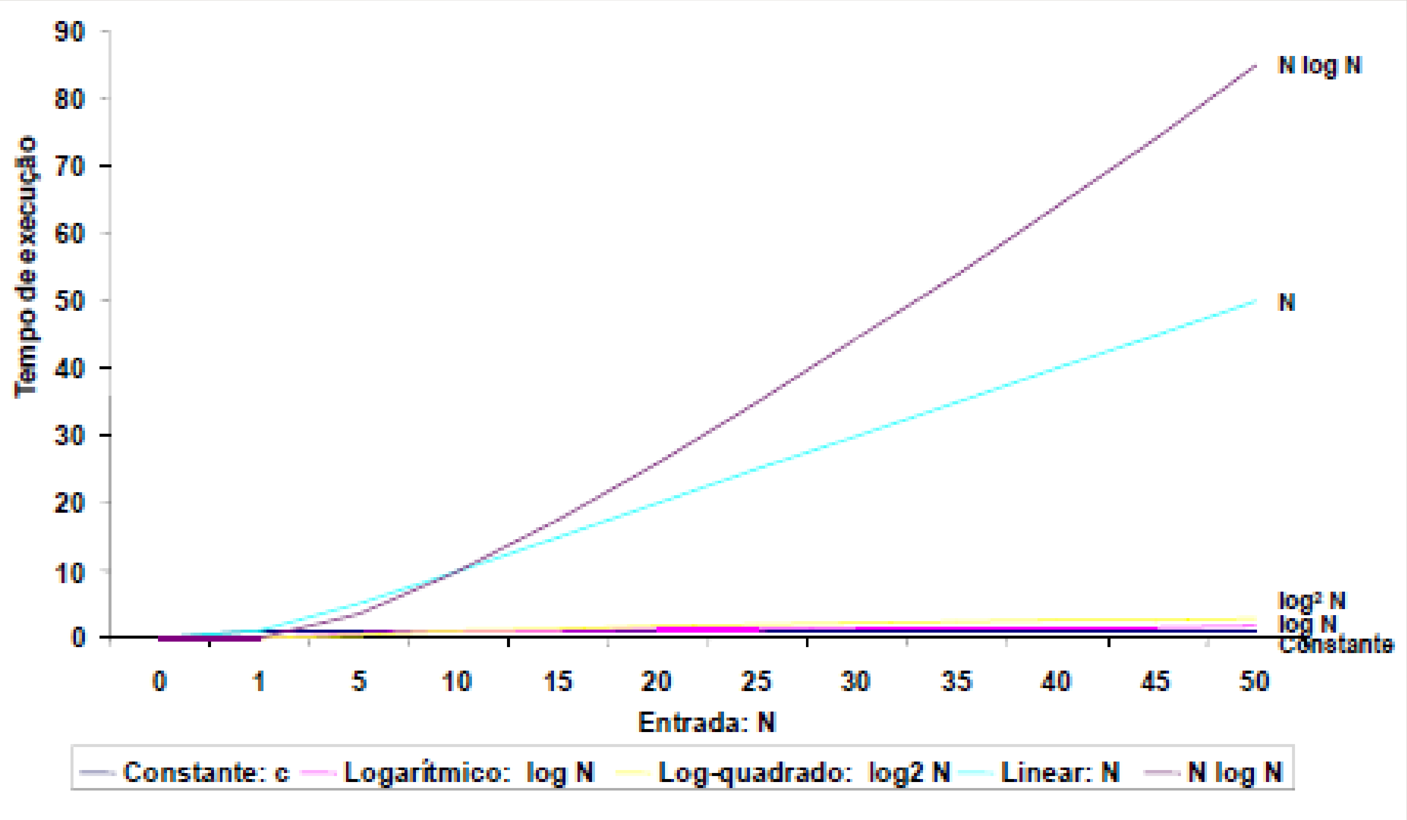
$5n^2 + n(\log n)^2 + 12$ $\in O(n^2)$ $\in O(n^3)$ $\notin O(n.\log^9 n)$

$\log n + 4$ $\in O(\log n)$ $\in O(n)$

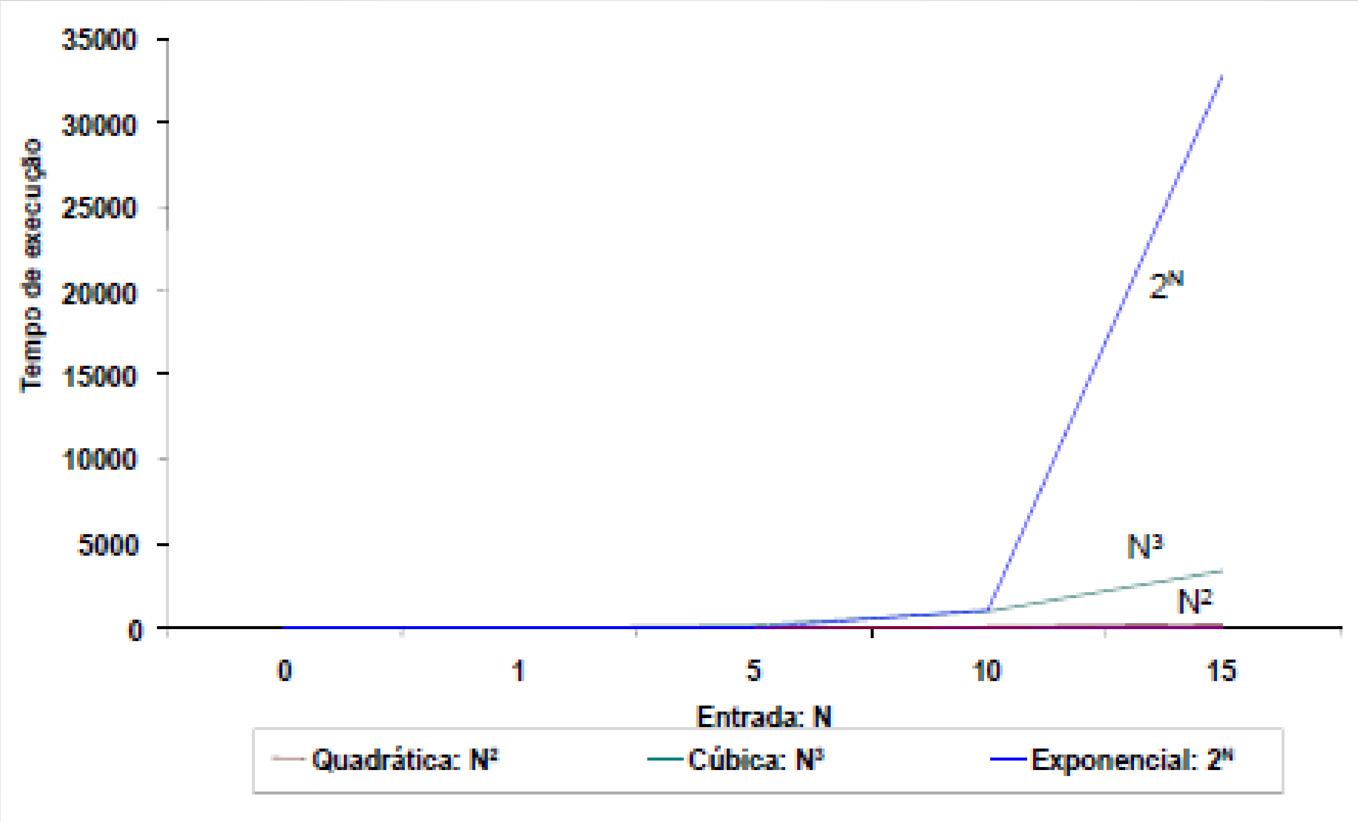
17 COMPARAÇÃO ENTRE FUNÇÕES



18 COMPARAÇÃO ENTRE FUNÇÕES



19 COMPARAÇÃO ENTRE FUNÇÕES



20 COMPARAÇÕES ENTRE FUNÇÕES

- A partir da notação O é possível estabelecer uma hierarquia entre as funções:

Constante	$O(1)$
Logarítmica	$O(\log n)$
Linear	$O(n)$
n.log n	$O(n.\log n)$
Quadrática	$O(n^2)$
Cúbica	$O(n^3)$
Polinomial	$O(n^k)$, com $k \geq 4$
Exponencial	$O(k^n)$, com $k > 1$

Maior ordem

- Evidentemente, as funções lineares, quadráticas e cúbicas também são polinomiais

ANÁLISE ASSINTÓTICA DE ALGORITMOS

- A análise assintótica de algoritmos descreve o tempo de execução em notação O
- Para realizar a análise assintótica de um algoritmo:
  - Calcula-se o número de operações primitivas executadas como função do tamanho da entrada.
  - Expressa-se esta função na notação O.
- Exemplo:
  - O algoritmo arrayMax executa no máximo 4n-1 operações primitivas.
  - Dizemos que o algoritmo arrayMax gasta tempo O(n), ou seja, tem complexidade linear.

NOTAÇÃO O NA ANÁLISE DE ALGORITMOS

- Em comandos consecutivos, somam-se os tempos:

```
for (int x=1; x < n; x++)
  <operação qualquer>;
```

$O(n)$

```
for (int x=1; x < n; x++)
  for (int y=1; y < n; y++)
    <operação qualquer>;
```

$O(n^2)$

- Tempo total: $O(n) + O(n^2) = O(n^2)$
- E se o for mais interno começasse a partir de x?

EXERCÍCIO

- Calcule o tempo de execução do Algoritmo do bubble sort em notação O

LIMITES INFERIORES

- Enquanto a notação O fornece limites superiores para o crescimento das funções, também há outras notações que oferecem mais informações interessantes.
- Seja $\Omega(g(n))$ o conjunto de funções $f(n)$ para as quais existem constantes positivas c e n_0 tais que $f(n) \geq c.g(n), \forall n \geq n_0$.
- A notação Ω fornece um limite inferior para o crescimento das funções.

- Na notação Ω convém utilizar a maior função possível

25 EXEMPLOS

$f(n) = 12n^2 - 10 \in \Omega(1)$

$f(n) = 12n^2 - 10 \in \Omega(n)$

$f(n) = 12n^2 - 10 \in \Omega(n^2)$

Entretanto, $f(n) = 12n^2 - 10 \notin \Omega(n^3)$

26 LIMITES INFERIORES E SUPERIORES

- Quando uma função pertence simultaneamente a O(g(n)) e a Ω(g(n)), dizemos que $f(n) \in \Theta(g(n))$

$$f(n) \in \Theta(g(n)) \iff f(n) \in (O(g(n)) \cap \Omega(g(n)))$$

$$f(n) \in \Theta(g(n)) \iff f(n) \in O(g(n)) \text{ e } g(n) \in O(f(n))$$

Mais precisamente, Θ(g(n)) é o conjunto de todas as funções para as quais existem constantes positivas c_1, c_2 e n_0 tais que:

$$c_1.g(n) \leq f(n) \leq c_2.g(n),\ \forall n \geq n_0$$

Basicamente, podemos dizer que $f(n)$ é $\Theta(g(n))$ se e somente se:

$$\lim_{n \to \infty} (f(n)/g(n)) = c, \text{ onde } c > 0$$

Por outro lado:

- Se $\lim_{n \to \infty} (f(n)/g(n)) = 1$, dizemos que $f(n) \sim g(n)$
- Se $\lim_{n \to \infty} (f(n)/g(n)) = 0$, dizemos que $f(n)$ é $o(g(n))$
- Se $\lim_{n \to \infty} (f(n)/g(n)) = \infty$, dizemos que $f(n)$ é $\omega(g(n))$

É possível fazer uma *analogia* entre a comparação assintótica de duas funções f e g e a comparação de dois números reais a e b.

$f(n) = O(g(n))$	$\approx$	$a \leq b$
$f(n) = \Omega(g(n))$	$\approx$	$a \geq b$
$f(n) = \Theta(g(n))$	$\approx$	$a = b$
$f(n) = o(g(n))$	$\approx$	$a < b$
$f(n) = \omega(g(n))$	$\approx$	$a > b$

$6n^4 + 12n^3 + 12$	$\in \Theta(n^4)$	$\in o(n^5)$	$\notin \omega(n^5)$
$3n^2 + 12n.\log n$	$\in \Theta(n^2)$	$\in \Omega(n)$	$\notin O(n.\log n)$
$5n^2 + n(\log n)^2 + 12$	$\in \Theta(n^2)$	$\in \omega(n.\log^2 n)$	$\notin \Omega(n^3)$
$\log n + 4$	$\in \Theta(\log n)$	$\notin o(\log n)$	

FIM DA AULA 6

Próxima aula:

Compressão de Dados