Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital

AULA 03

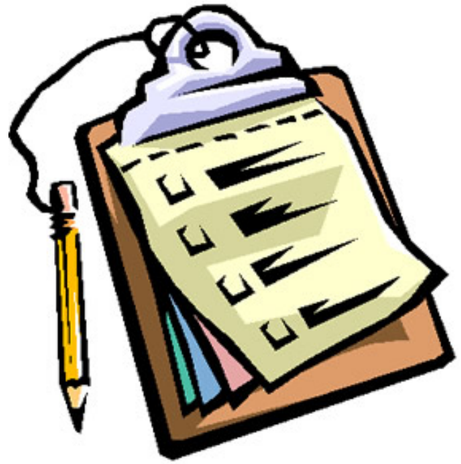
Estrutura de dados básico I (EDB1)

Prof. Msc. Janiheryson Felipe (Felipe)

Natal, RN
2023

OBJETIVOS DA AULA

- Apresentar as principais notações assintóticas:
  - Notação Big O (O):
  - Notação Ômega (Ω):
  - Notação Theta (Θ):
  - Resolver exercícios relacionados à temática.



NOTAÇÃO BIG O (O)

A notação **Big O** é usada para descrever a complexidade assintótica superior de um algoritmo, ou seja, o limite superior do tempo de execução. Em outras palavras, a notação **O** descreve quanto tempo um algoritmo leva para resolver um problema, no pior caso possível.

NOTAÇÃO BIG O (O)

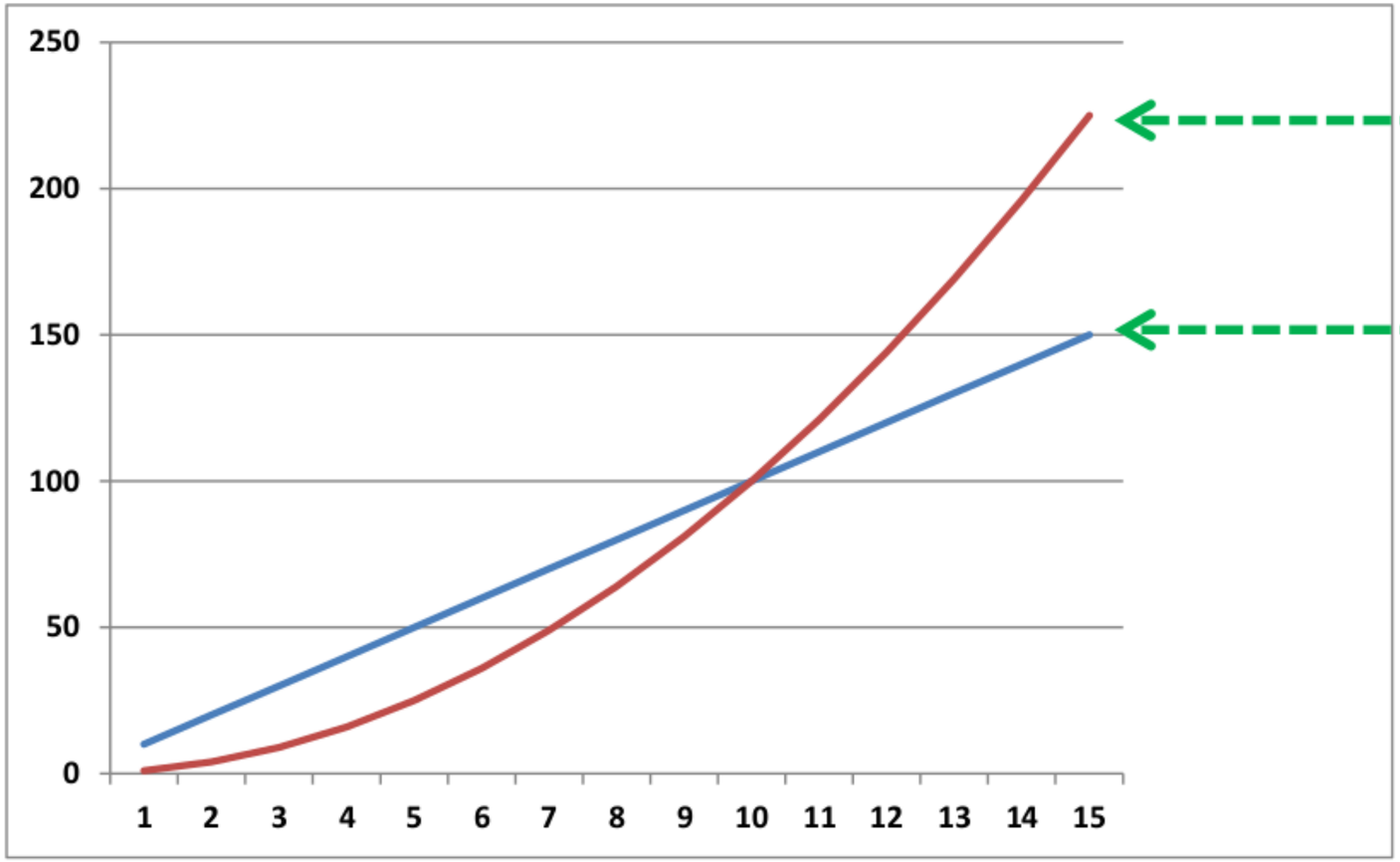
Quando escrevemos f(n) = O(g(n)), significa que a função f cresce no máximo tão rapidamente quanto à função g, ou seja, f é limitada superiormente por g, a partir de um determinado valor n0.

Isso é expresso matematicamente como f(n) <= c * g(n), para todos os valores de n maiores ou iguais a n0, onde c é uma constante positiva.

NOTAÇÃO BIG O (O)

Por exemplo, se um algoritmo de ordenação tem complexidade $O(n^2)$, isso significa que o tempo de execução do algoritmo cresce proporcionalmente ao quadrado do tamanho da entrada.

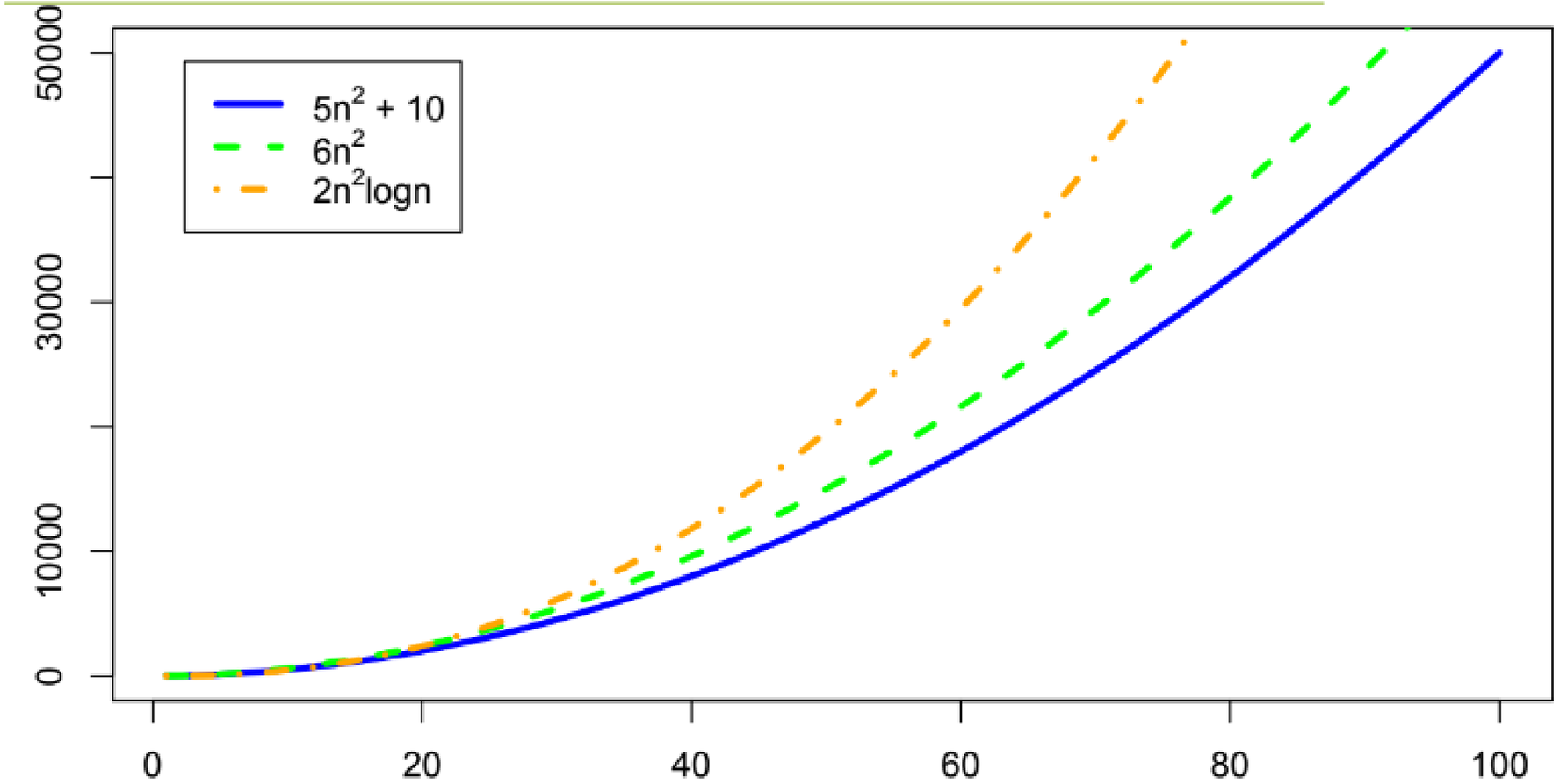
NOTAÇÃO BIG O (O)



$g(x) = x^2$

$f(x) = 10x$

NOTAÇÃO BIG O (O)



NOTAÇÃO OMÊGA (Ω)

A notação Ω é usada para descrever a complexidade assintótica inferior de um algoritmo, isto é, o limite inferior do tempo de execução. Em outras palavras, a notação Ω descreve quanto tempo um algoritmo leva para resolver um problema, no melhor caso possível.

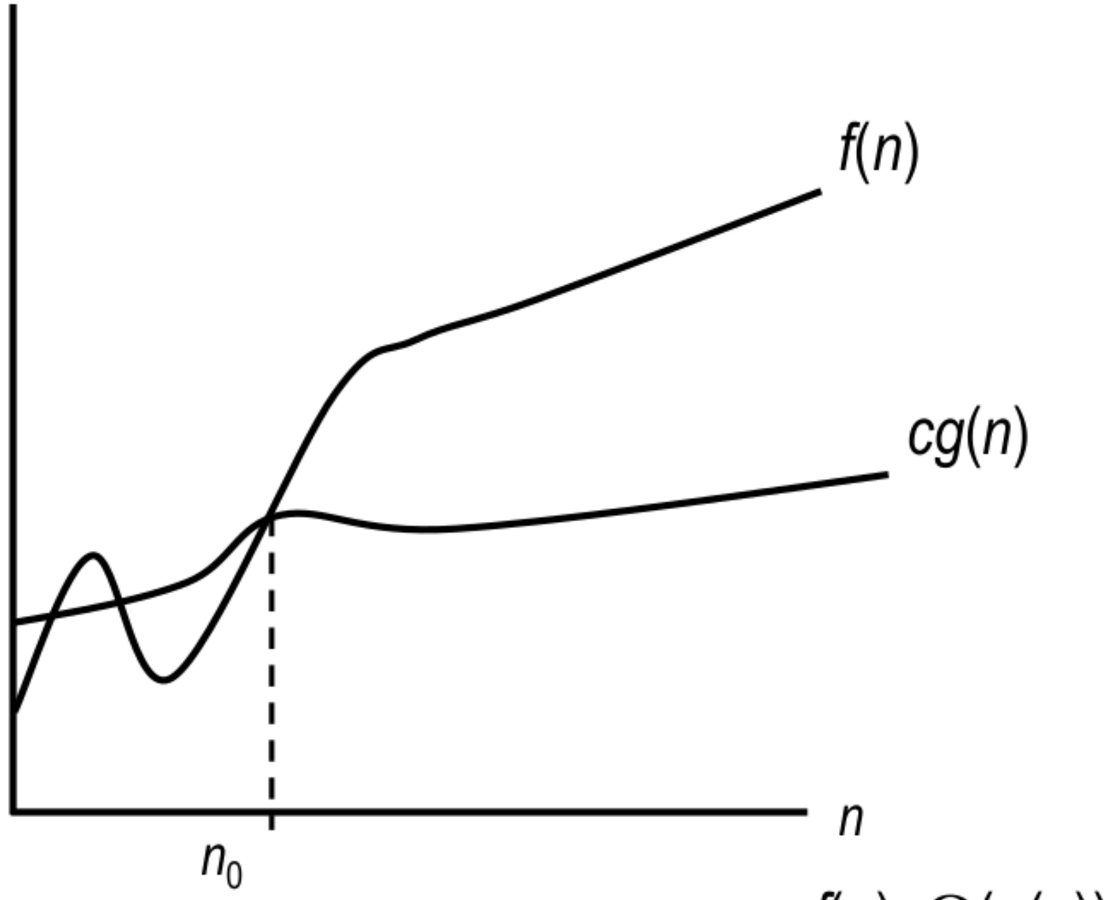
NOTAÇÃO ÔMEGA (Ω)

Quando escrevemos f(n) = Ω(g(n)), significa que a função f cresce no mínimo tão rapidamente quanto a função g, ou seja, f é limitada inferiormente por g, a partir de um determinado valor n0. Isso é expresso matematicamente como f(n) >= c * g(n), para todos os valores de n maiores ou iguais a n0, onde c é uma constante positiva.

NOTAÇÃO ÔMEGA (Ω)

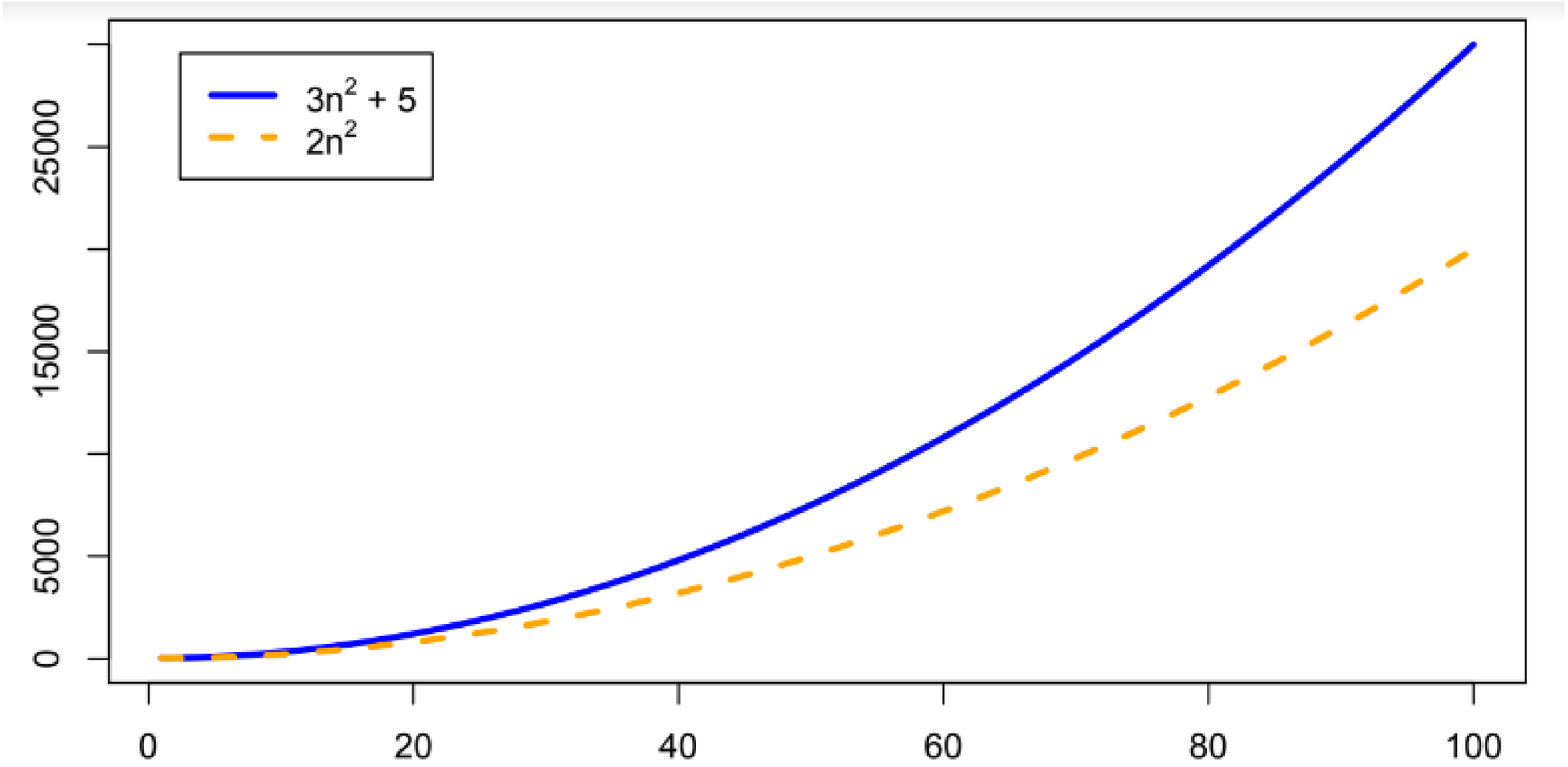
Por exemplo, se um algoritmo de ordenação tem complexidade **Ω(n)**, isso significa que o tempo de execução do algoritmo cresce proporcionalmente ao tamanho da entrada.

NOTAÇÃO ÔMEGA (Ω)



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

NOTAÇÃO ÔMEGA (Ω)



NOTAÇÃO THETA (Θ)

A notação **Θ** é usada para descrever a complexidade assintótica exata de um algoritmo, isto é, o tempo de execução exato em todos os casos possíveis. Em outras palavras, a notação **Θ** descreve quanto tempo um algoritmo leva para resolver um problema, no melhor e no pior caso possível.

NOTAÇÃO THETA (Θ)

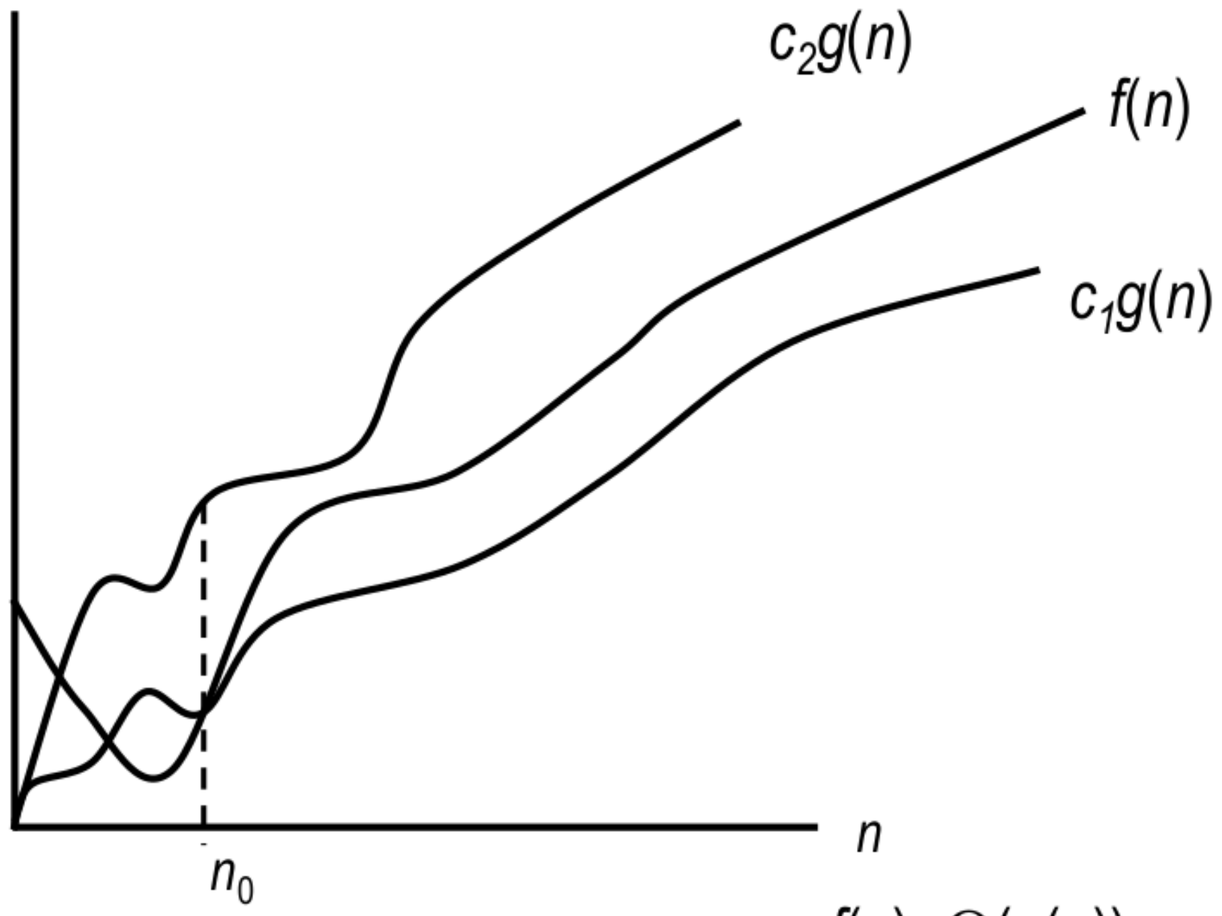
Quando escrevemos f(n) = Θ(g(n)), significa que a função f cresce na mesma taxa que a função g, ou seja, f é limitada tanto superior quanto inferiormente por g, a partir de um determinado valor n0.

Isso é expresso matematicamente como c1 * g(n) <= f(n) <= c2 * g(n), para todos os valores de n maiores ou iguais a n0, onde c1 e c2 são constantes positivas.

NOTAÇÃO THETA (Θ)

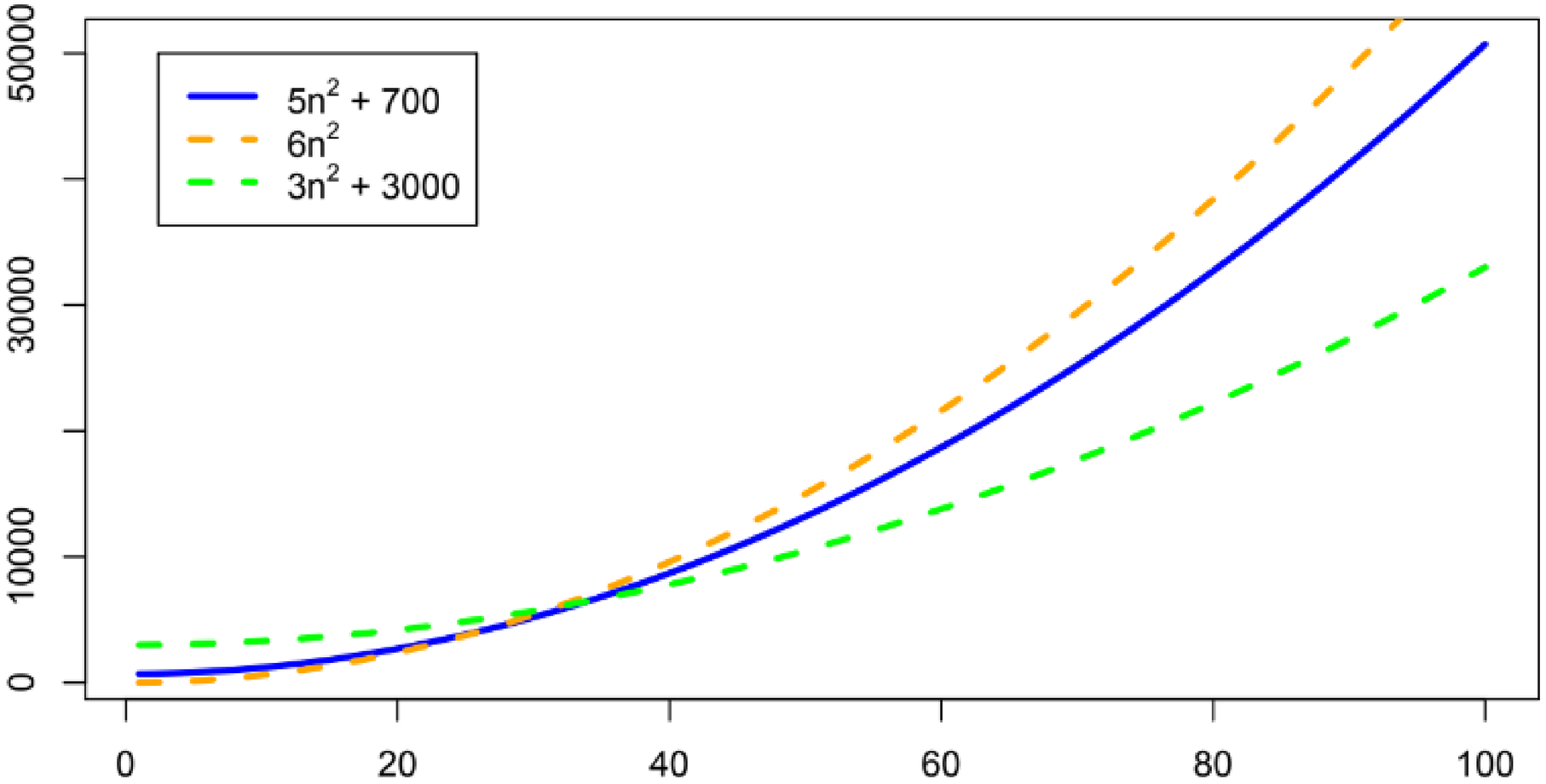
Por exemplo, se um algoritmo de busca binária tem complexidade **Θ**(log n), isso significa que o tempo de execução do algoritmo cresce proporcionalmente ao logaritmo do tamanho da entrada.

NOTAÇÃO THETA (Θ)

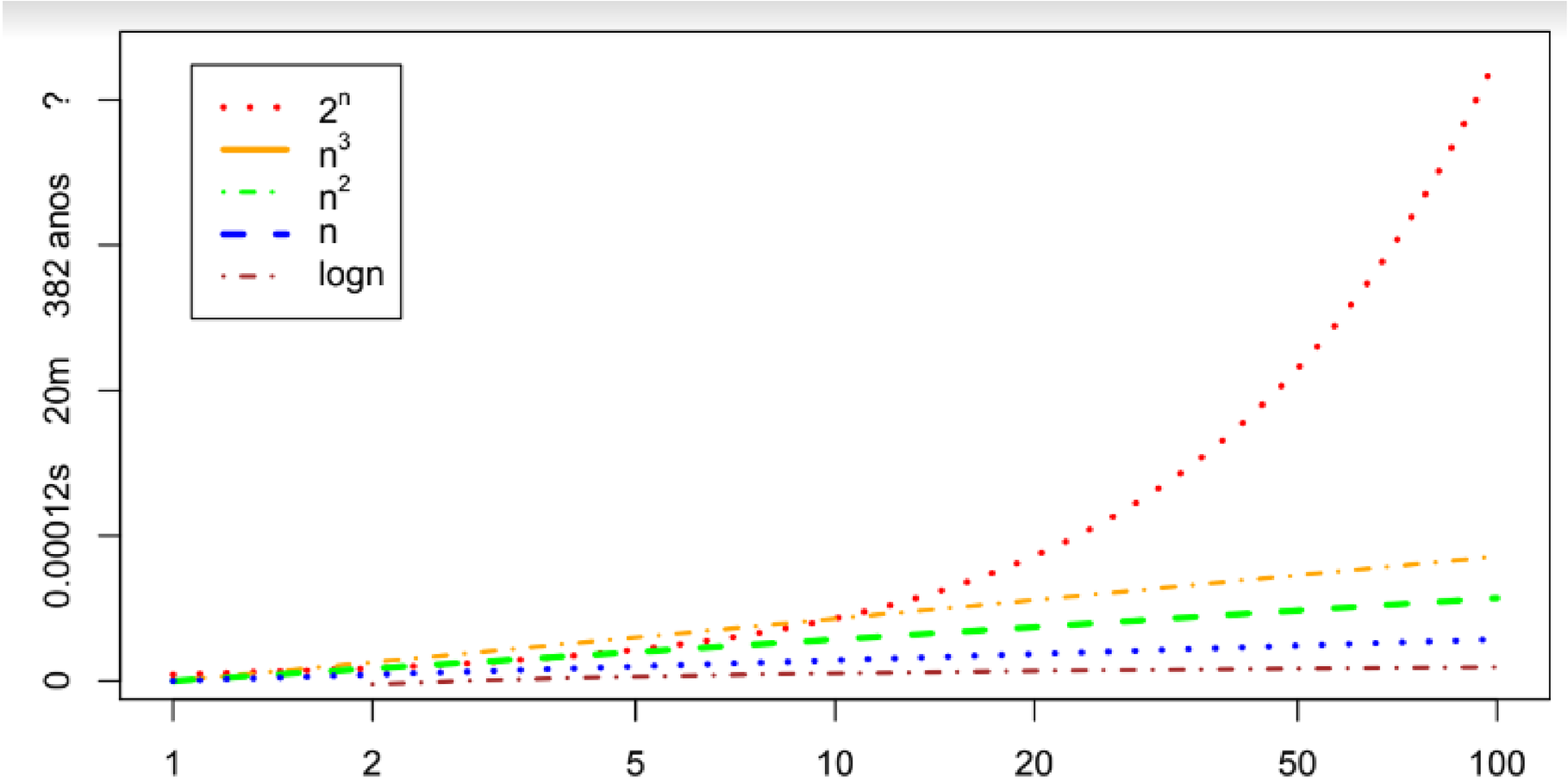


$f(n)=\Theta(g(n))$

COMPARANDO FUNÇÕES



COMPARANDO FUNÇÕES



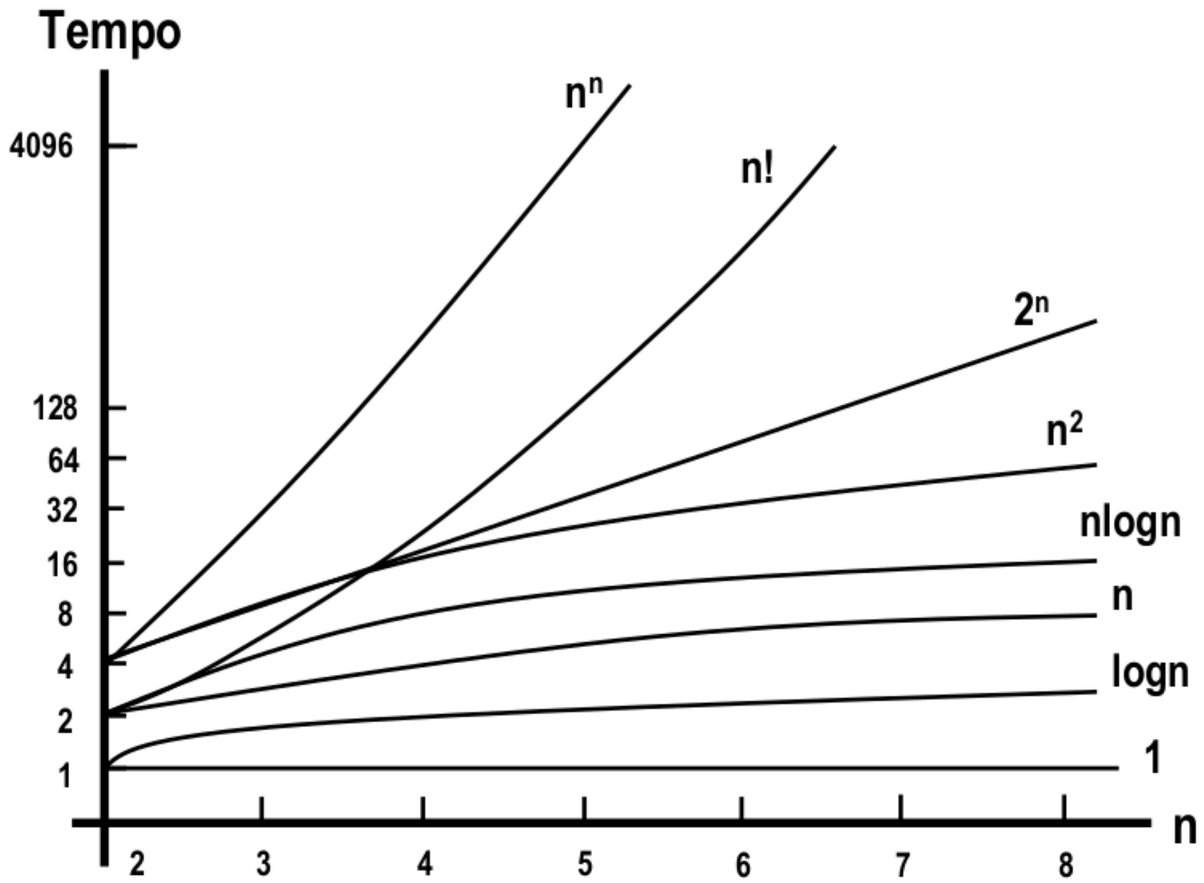
COMPARANDO FUNÇÕES

Melhor

Pior

- $O(1)$ constante
- $O(\log n)$ logarítmica
- $O(n)$ linear
- $O(n \log n)$ log linear
- $O(n^2)$ quadrática
- $O(n^3)$ cúbica
- $O(2^n)$ exponencial

COMPARANDO FUNÇÕES



COMPARANDO FUNÇÕES

n	A_1 $T(n)=n$	A_2 $T(n)=n\log n$	A_3 $T(n)=n^2$	A_4 $T(n)=n^3$	A_5 $T(n)=2^n$
16	0,016s	0,064s	0,256s	4s	1m4s
32	0,032s	0,16s	1s	33s	46d
512	0,512s	9s	4m22s	1d13h	10^{137} Séculos

COMPARE OS ALGORITMOS DE ORDENAÇÃO

EXERCÍCIO 1

Expression	Dominant term(s)	$O(\ldots)$
$5 + 0.001n^3 + 0.025n$		
$500n + 100n^{1.5} + 50n \log_{10} n$		
$0.3n + 5n^{1.5} + 2.5 \cdot n^{1.75}$		
$n^2 \log_2 n + n(\log_2 n)^2$		
$n \log_3 n + n \log_2 n$		
$3 \log_8 n + \log_2 \log_2 \log_2 n$		
$100n + 0.01n^2$		
$0.01n + 100n^2$		
$2n + n^{0.5} + 0.5n^{1.25}$		
$0.01n \log_2 n + n(\log_2 n)^2$		
$100n \log_3 n + n^3 + 100n$		
$0.003 \log_4 n + \log_2 \log_2 n$		

EXERCÍCIO 2

Os algoritmos A e B gastam exatamente $Ta(n) = 0,1n^2log10n$ and $Tb(n) = 2,5n^2$ ms respectivamente, para um problema de tamanho n. Escolha o algoritmo, que tem melhor desempenho na notação Big-O

DÚVIDAS