Introdução à Complexidade Computacional - Parte 2

Algoritmos e Programação 2
Prof. Dr. Anderson Bessa da Costa
Universidade Federal de Mato Grosso do Sul

Relembrando …

- Ordem de crescimento provê caracterização eficiência do algoritmo
  - Permite comparar o desempenho relativo de algoritmos
- *Merge sort* no pior caso $\Theta(n \lg n)$
- *Insertion sort* no pior caso $\Theta(n^2)$
  - *Merge sort* é melhor que insertion sort (entrada n grande)
- Podemos determinar o tempo exato de execução do algoritmo
  - Não vale a pena
  - Para entradas grandes o suficiente, as constantes multiplicativas e termos de baixa ordem serão dominados

Introdução

- Entradas grandes o suficiente
- Apenas ordem de crescimento é relevante
- Nós estamos estudando a eficiência **assintótica** dos algoritmos
  - Estamos interessados em como o tempo de execução de um algoritmo aumenta com o tamanho da entrada **no limite**
    - Entrada cresce sem limite
- Geralmente, um algoritmo que é assintóticamente mais eficiente será a melhor escolha, exceto para pequenas entradas

Complexidade de Algoritmos

- É possível determinar tempo de execução por **métodos empíricos**
  - Obter o tempo de execução através da execução propriamente dita
- Em contrapartida, é possível obter uma ordem de grandeza do tempo de execução através de **métodos analíticos**
  - Objetivo é determinar expressão matemática que traduza o comportamento de tempo de um algoritmo
  - Visa aferir o tempo de execução de forma independente do computador utilizando, linguagem e dos compiladores empregados

Expressão Matemática

As seguintes simplificações serão introduzidas:

- Suponha quantidade de dados a serem manipulados seja suficientemente grande. Somente o comportamento assintótico será avaliado, ou seja, a expressão matemática fornecerá valores de tempo que são válidos quando a quantidade de dados correspondente crescer o suficiente
- Não serão consideradas constantes aditivas ou multiplicativas na expressão matemática obtida

Noção de Complexidade

- As definições de complexidade implicam o atendimento das duas simplificações
- Por exemplo, valor de número de passos igual a $3n$ será aproximado para n
- Além disso, como o interesse é restrito a valores assintóticos, termos de menor grau também podem ser desprezados. Assim, um valor de números de passos igual a $n^2 + n$ será aproximado para n^2
- O valor $6n^3 + 4n - 9$ será transformado em n^3

Noção de Complexidade (cont.)

- O que nos interessa de fato é a **taxa de crescimento**, ou **ordem de crescimento**, do tempo de execução
- Então, nós consideramos apenas o termo principal da fórmula (e.g., an^2), uma vez que os termos de menor ordem são relativamente insignificantes para n grande
- Da mesma forma é por isso que eliminamos as constante multiplicando o termo principal

Complexidade Assintótica

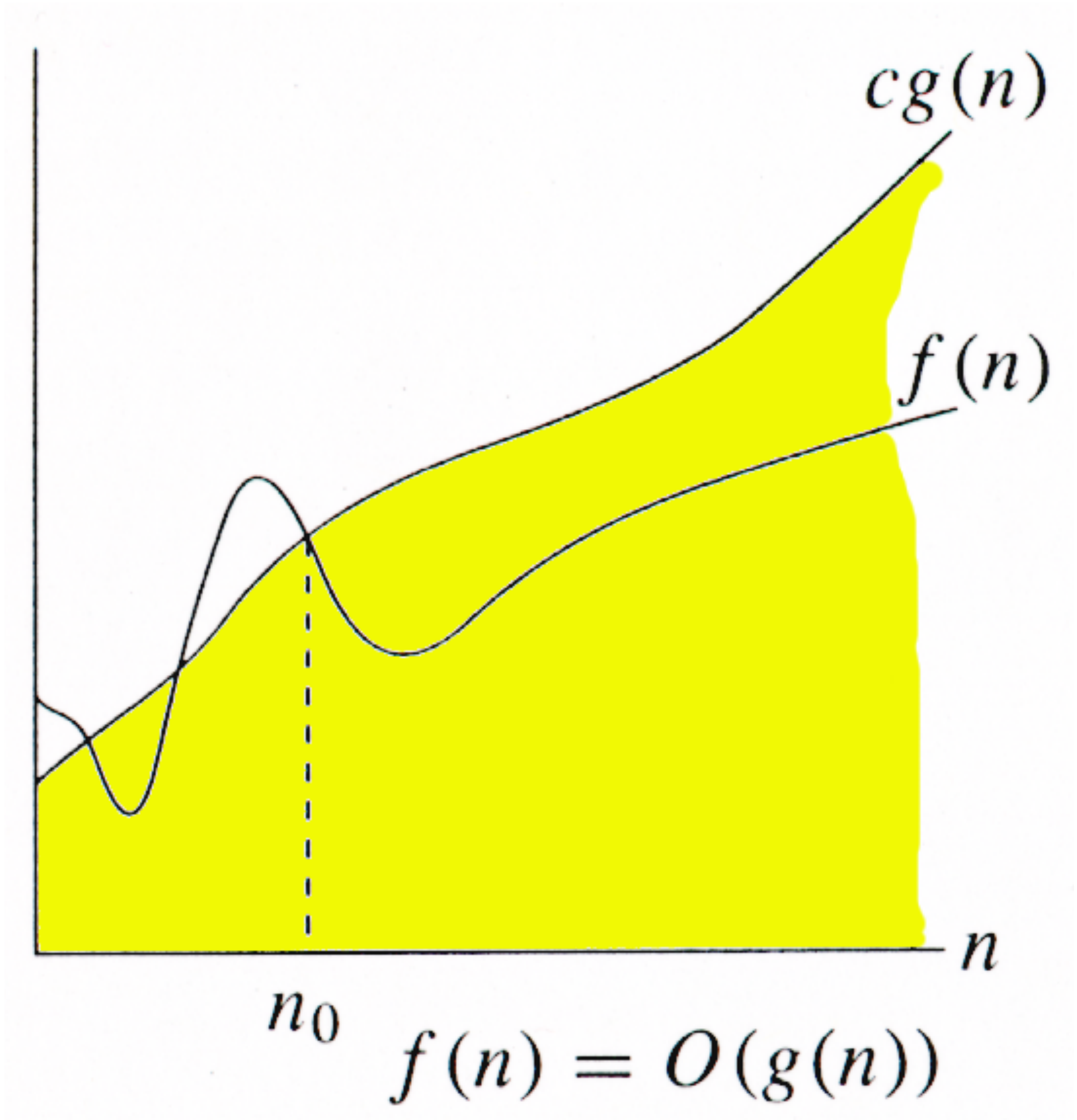
- Torna-se útil, portanto, descrever operadores matemáticos que sejam capazes de representar situações como essa
- As notações O, Ω e Θ serão utilizadas com essa finalidade

Notação O

- Sejam f, g funções reais positivas de variável inteira n. Diz-se que $f(n)$ é $O(g(n))$, escrevendo-se $f(n) = O(g(n))$, quando existir uma constante $c > 0$ e um valor inteiro n_0, tal que

  $$n \geq n_0 \Rightarrow 0 \leq f(n) \leq c \cdot g(n)$$

- Ou seja, a função $g(n)$ atua como um **limite superior assintótico**



Notação O (cont.)

- Escrevemos $f(n) = O(g(n))$ para indicar que a função $f(n)$ é um membro do conjunto $O(g(n))$
- Quando $a > 0$, qualquer função linear $an + b$ é $O(n^2)$
    - $c = a + |b|$
    - $n_0 = max(1, -b/a)$
- Na literatura, a notação O é muitas vezes utilizada informalmente para descrever limites assintóticos "justos"
    - Pode parecer estranho escrever $n = O(n^2)$

Notação O: Exemplos

- $f(n) = n^2 - 1 \Rightarrow f(n) = O(n^2)$.
- $f(n) = n^2 - 1 \Rightarrow f(n) = O(n^3)$.
- $f(n) = 403 \Rightarrow f(n) = O(1)$.
- $f(n) = 5 + 2\log n + 3\log^2 n \Rightarrow f(n) = O(\log^2 n)$.
- $f(n) = 5 + 2\log n + 3\log^2 n \Rightarrow f(n) = O(n)$.
- $f(n) = 3n + 5\log n + 2 \Rightarrow f(n) = O(n)$.
- $f(n) = 5 \cdot 2^n + 5n^{10} \Rightarrow f(n) = O(2^n)$.

Notação O: Ordenação por Inserção

- A notação O será utilizado para exprimir complexidades
- Por exemplo, no **melhor caso** do algoritmo **ordenação por inserção** efetua $(c_1 + c_2 + c_3 + c_4 + c_7) \times n - (c_2 + c_3 + c_4 + c_7)$ passos. Logo, a sua complexidade é $O(n)$
- No **pior caso**:

$$(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2})n^2 + (c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8)n - (c_2 + c_4 + c_5 + c_8)$$

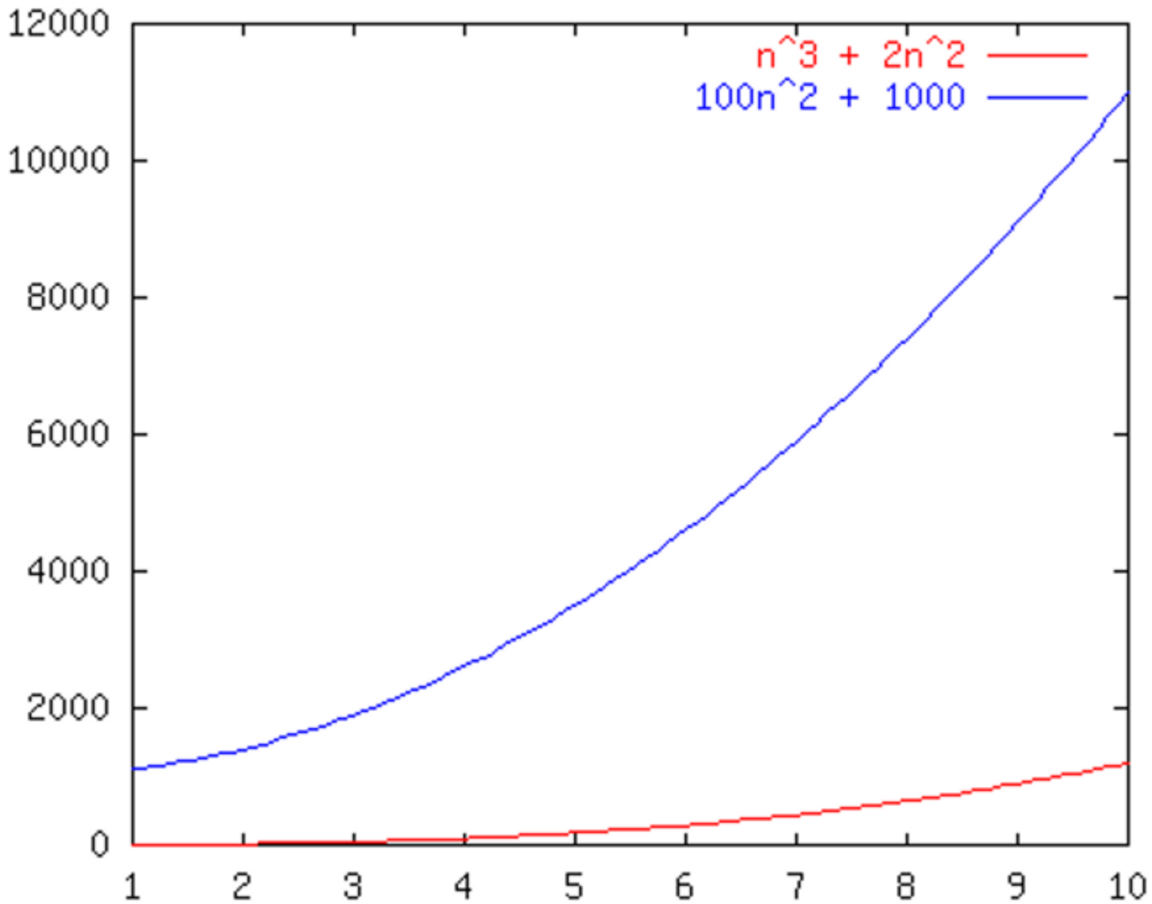
passos. Logo, a sua complexidade é $O(n^2)$

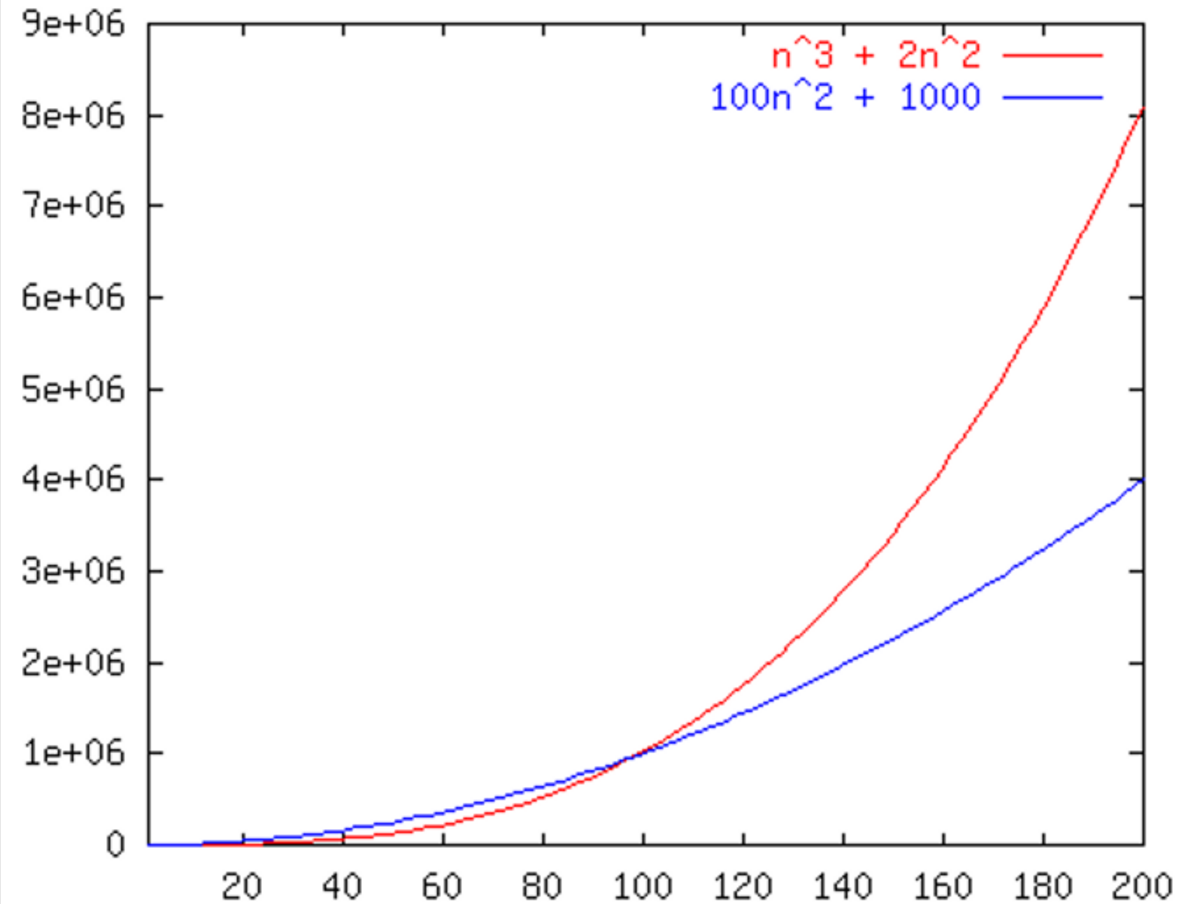
Notação O: Mostre que

- Mostre que $3n^3 = O(n^4)$
- **Resposta:** Segue que $O(g(n)) = \{f(n)$: existem constantes positiva c e n_0, tal que $0 \leq f(n) \leq cg(n)$ para todo $n \geq n_0\}$. Seja $g(n) = n^4$ e $f(n) = 3n^3$, para $c = 4$ e $n_0 = 1$, temos que $0 \leq 3n^3 \leq 4n^4$ é verdade para todo $n \geq n_0$.

Corrida 1

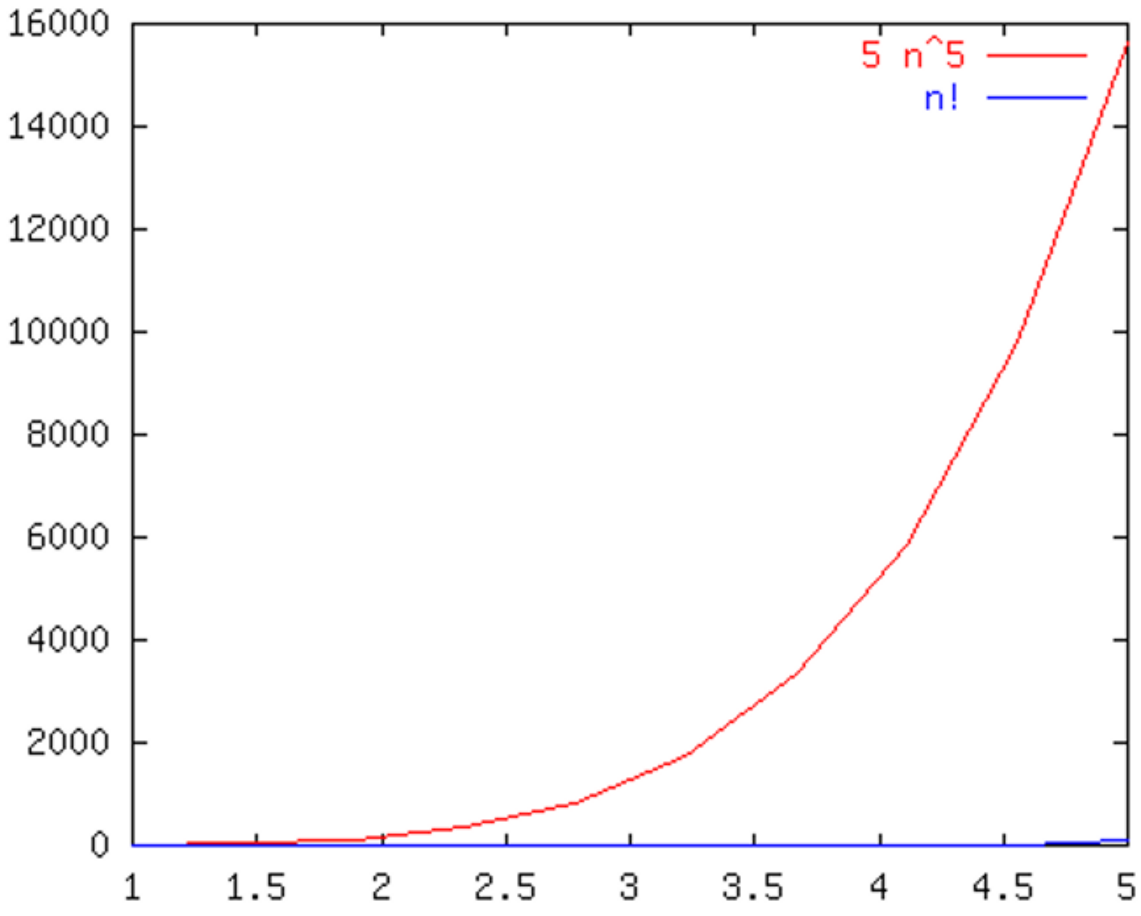
$\mathbf{n^3 + 2n^2}$ vs. $\mathbf{100n^2 + 1000}$

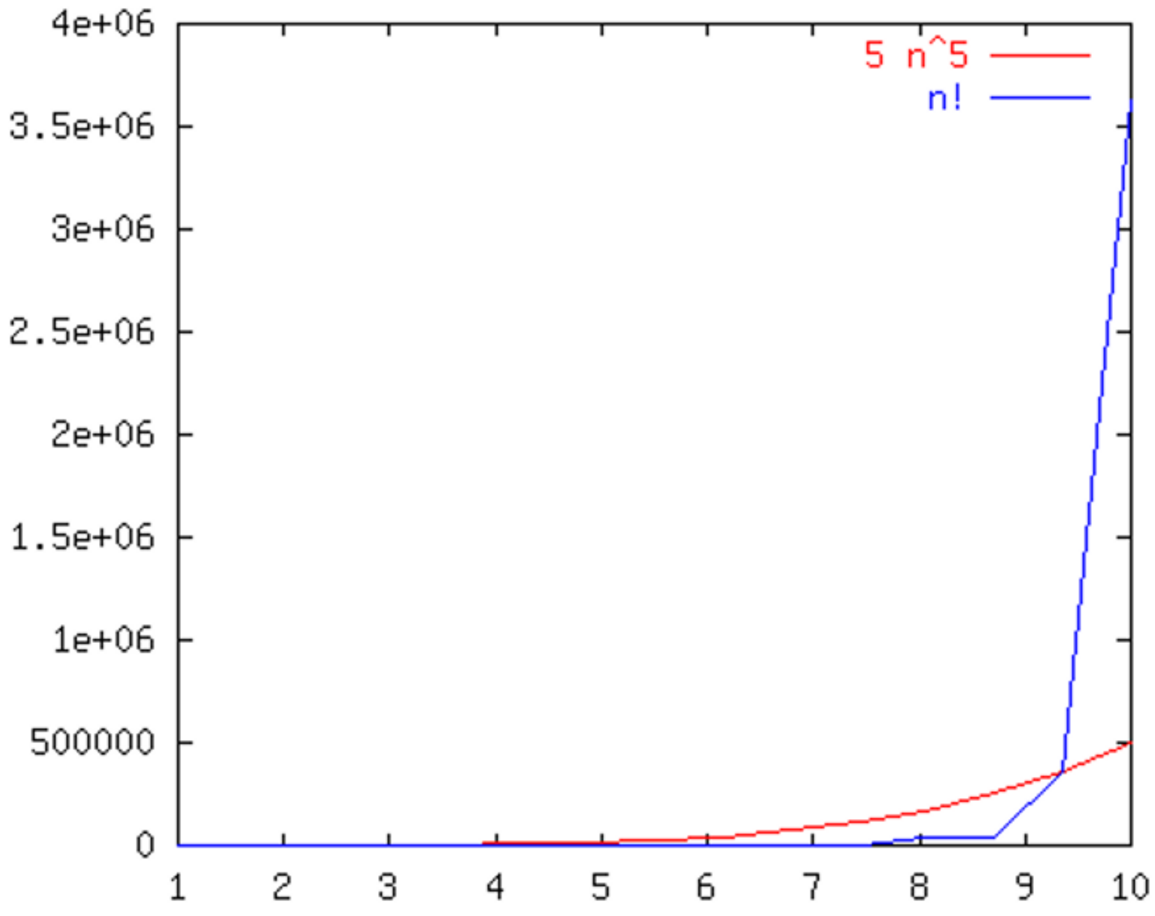




Corrida 2

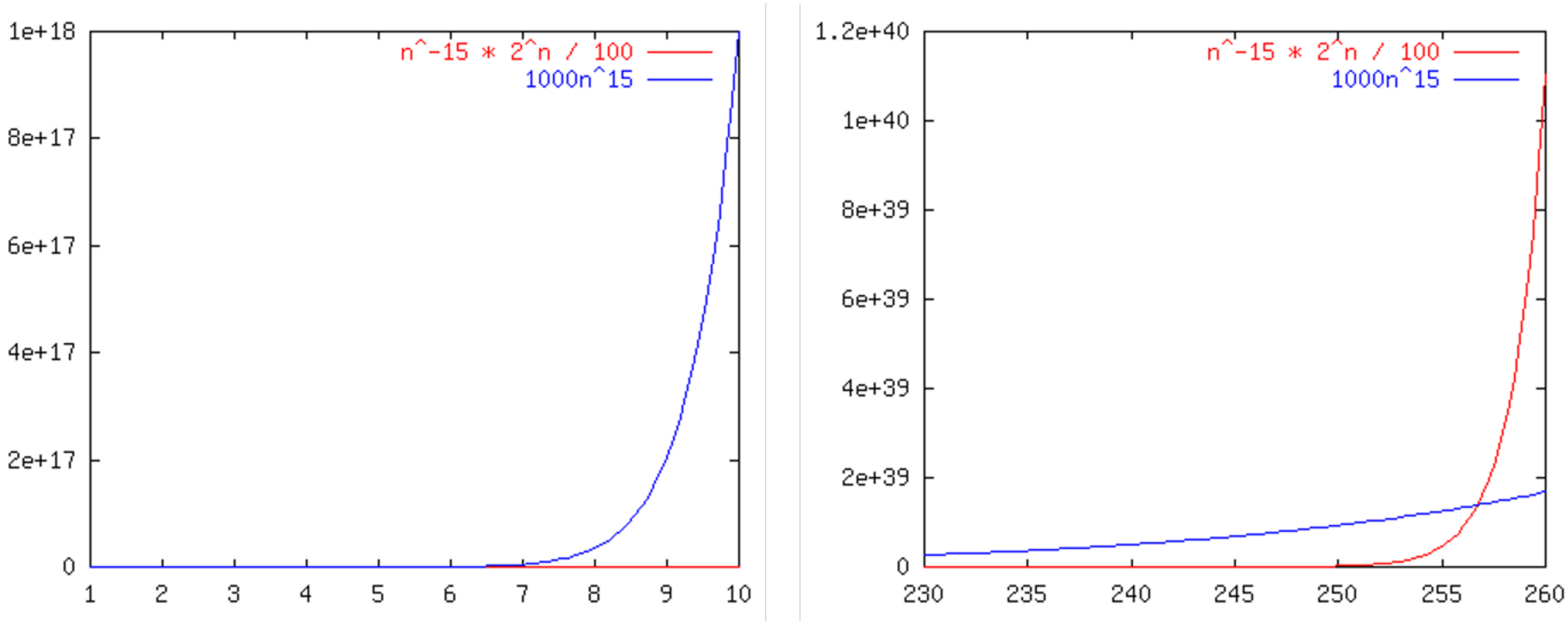
$5n^5$ vs. **n!**





Corrida 3

$\mathbf{n^{-15}2^n/100}$ vs. $\mathbf{1000n^{15}}$

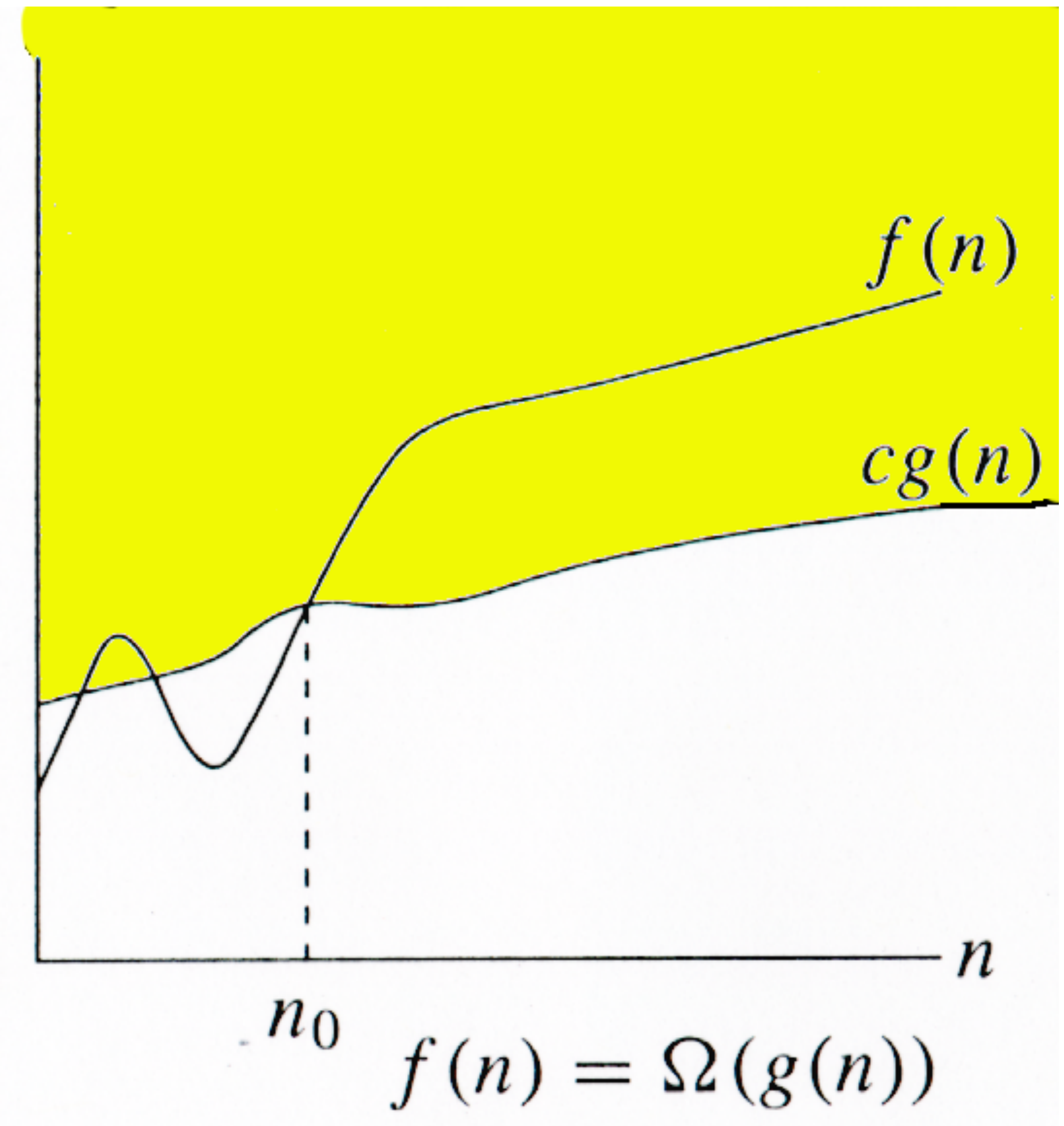


Notação Ω

- Sejam f, g funções reais positivas de variável inteira n. Diz-se que $f(n)$ é $\Omega(g)$, escrevendo-se $f(n) = \Omega(g(n))$, quando existir uma constante $c > 0$ e um valor inteiro n_0, tal que

  $$n \geq n_0 \Rightarrow 0 \leq cg(n) \leq f(n).$$

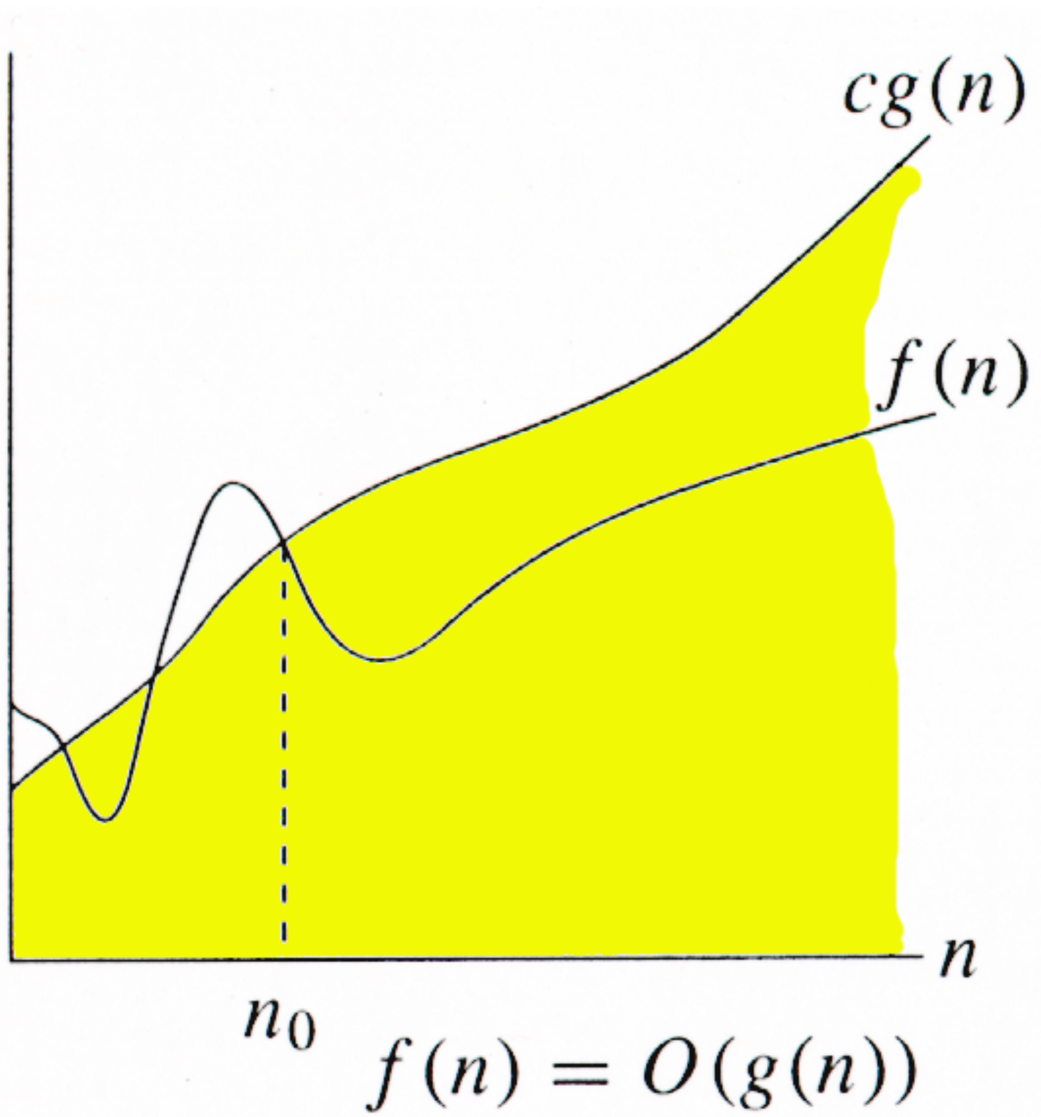
- Ou seja, a função $g(n)$ atua como um **limite inferior assintótico**

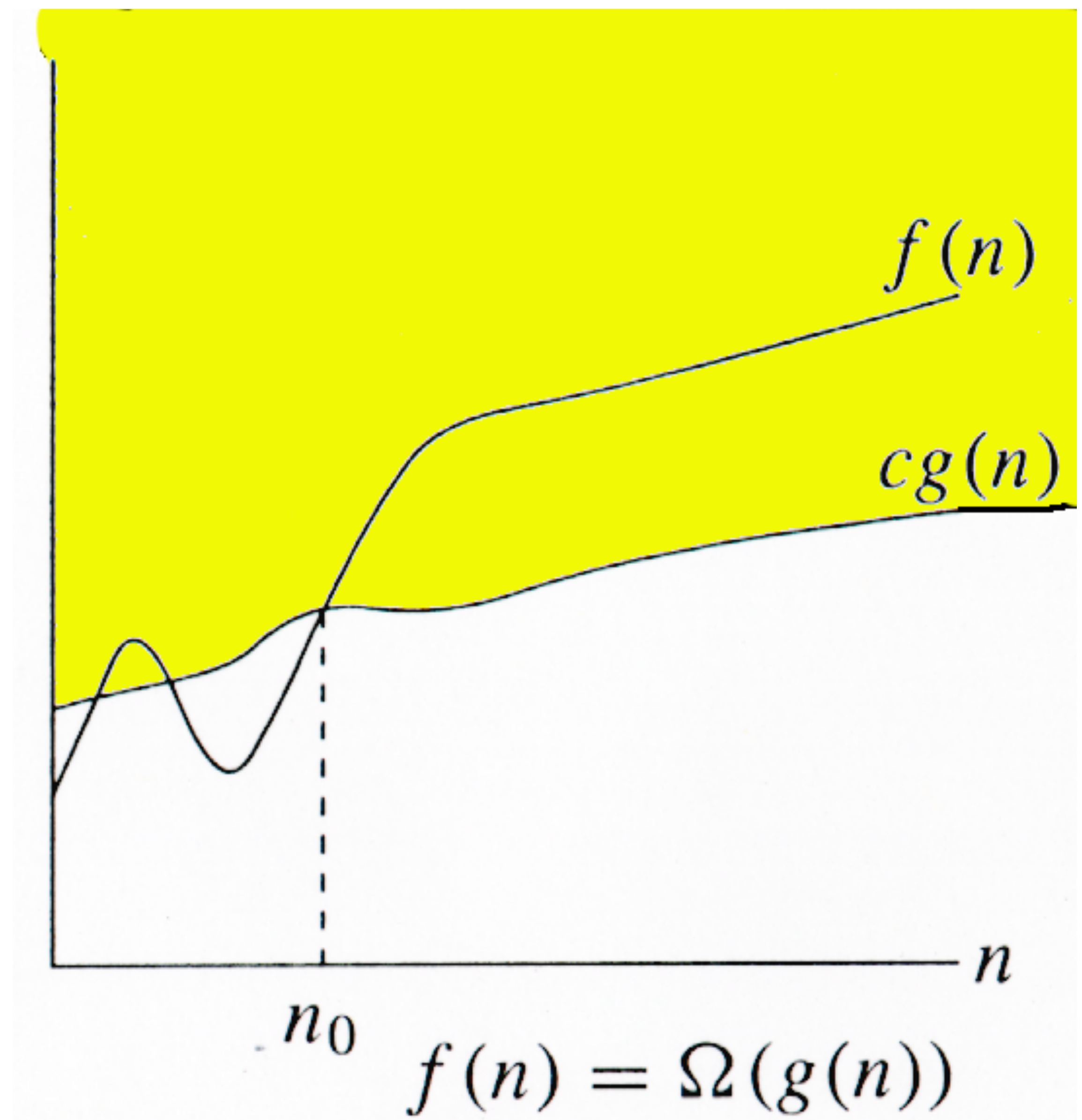


Notação Ω (cont.)

- Por exemplo, se $f(n) = n^2 - 1$, então
  - são válidas as igualdades $f = \Omega(n^2)$, $f = \Omega(n)$ e $f = \Omega(1)$
  - mas não $f(n) = \Omega(n^3)$

Notação O x Ω





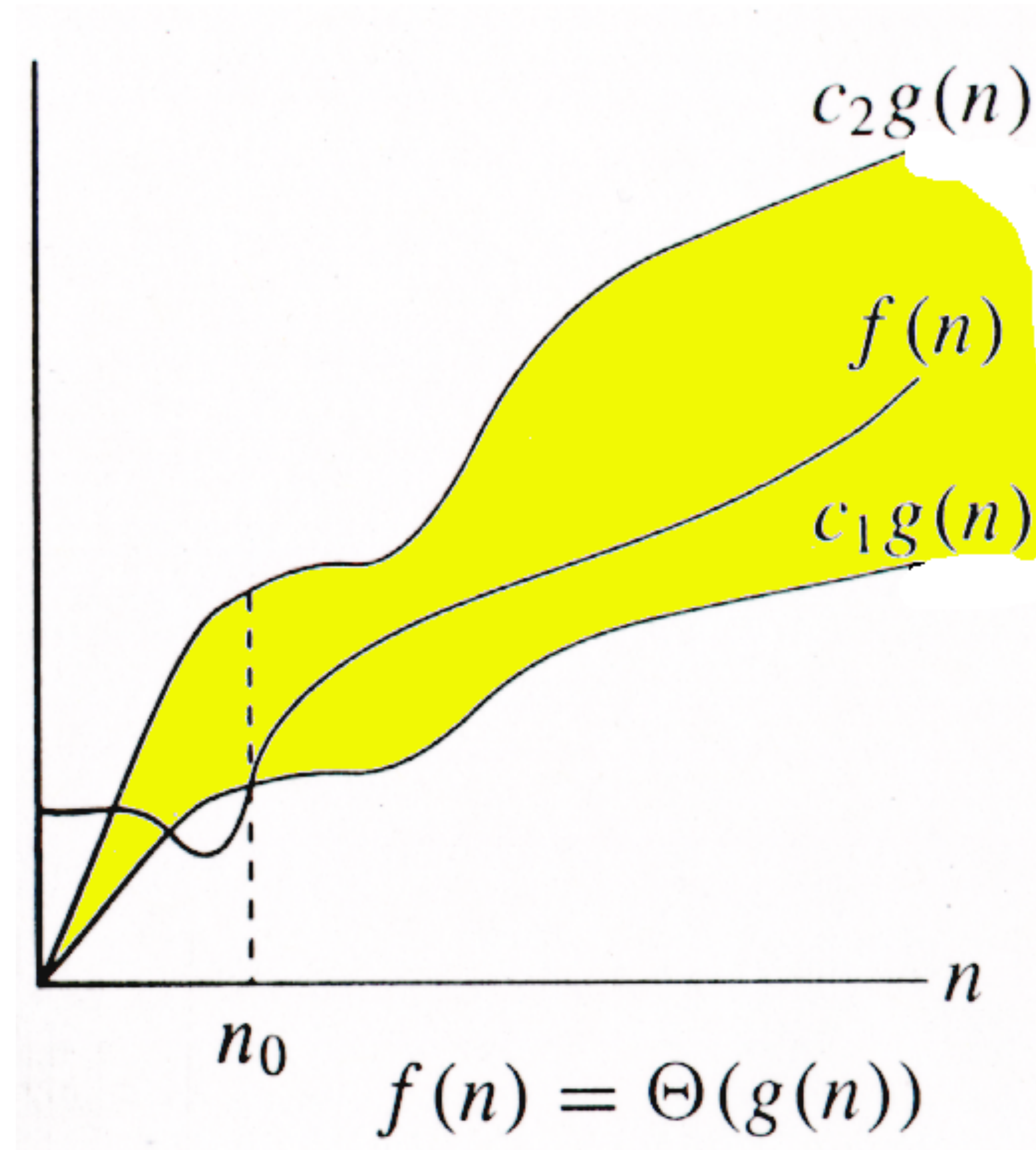
Notação Ω: Mostre que

- Mostre que $n^2 - 1 = \Omega(n^2)$
- **Resposta:** Segue que $\Omega(g(n)) = \{f(n)$: existem constantes positiva c e n_0, tal que $0 \leq cg(n) \leq f(n)$ para todo $n \geq n_0\}$. Seja $g(n) = n^2$ e $f(n) = n^2 - 1$, para $c = 1/2$ e $n_0 = 2$, temos que $\frac{1}{2}n^2 \leq n^2 - 1$ é verdade para todo $n \geq n_0$.

Notação Θ

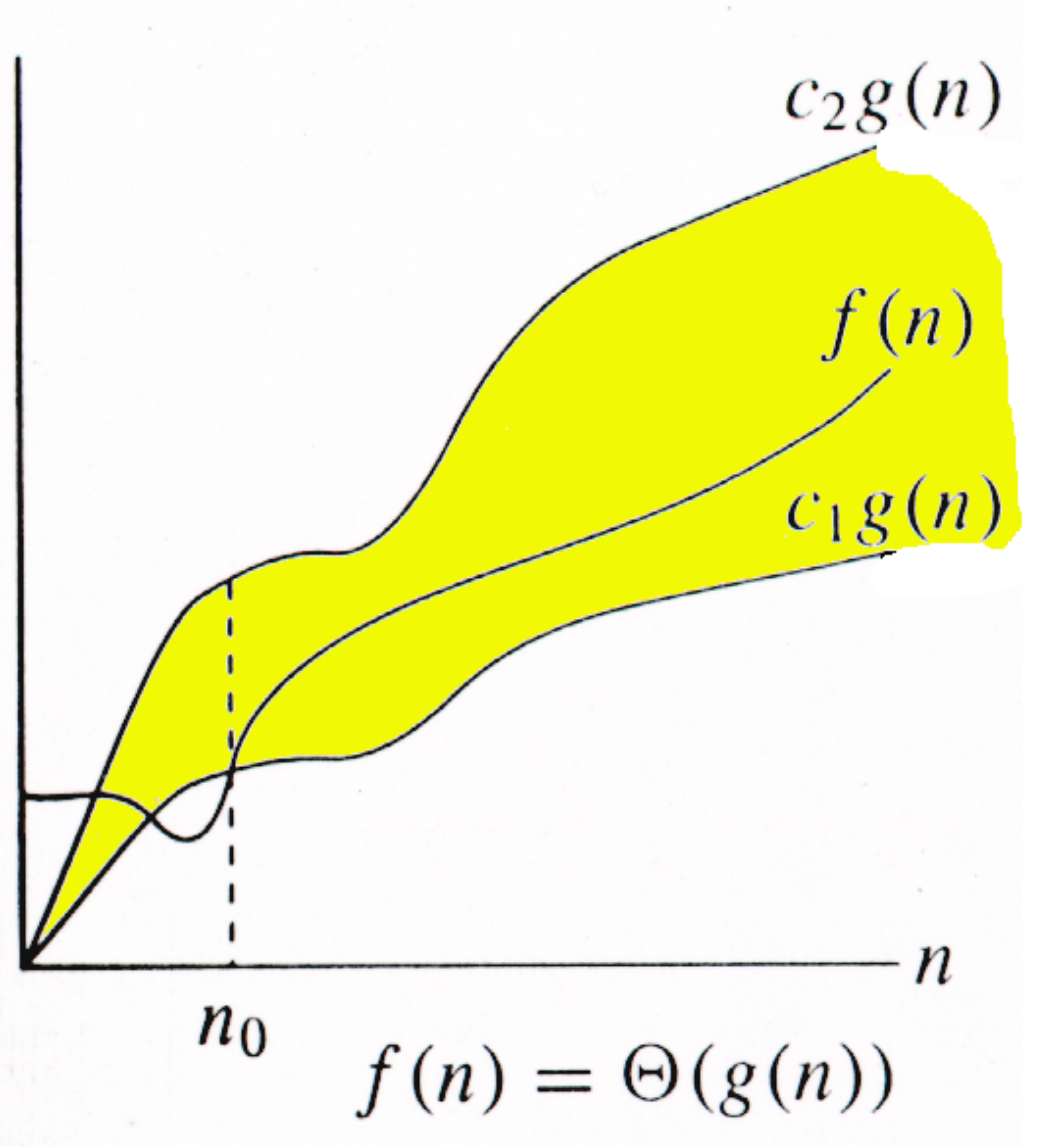
- Dado uma função $g(n)$, nós denotamos por $\Theta(g(n))$ o conjunto de funções

  $\Theta(g(n)) = \{f(n)$: existem constantes positivas c_1, c_2 e n_0 tal que $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para todo $n \geq n_0\}$



Notação Θ (cont.)

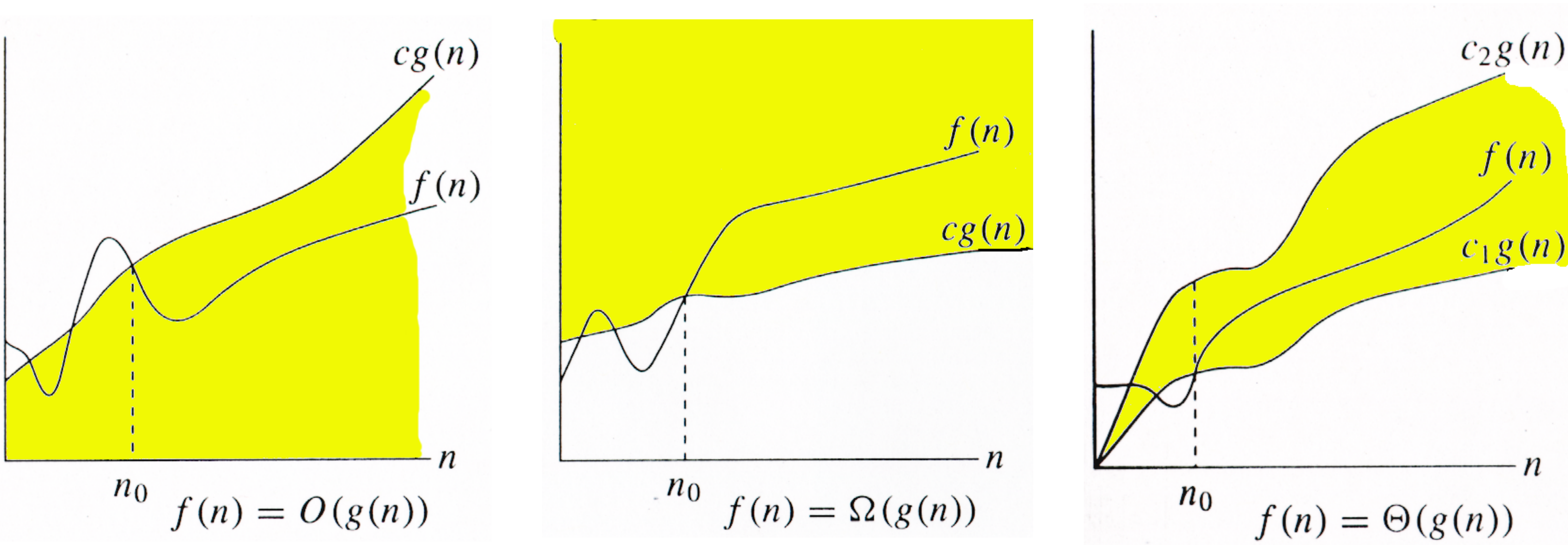
- Sejam f, g funções reais positivas da variável inteira n. Diz-se que $f(n)$ é $\Theta(g(n))$, escrevendo-se $f(n) = \Theta(g(n))$, quando ambas as condições $f(n) = O(g(n))$ e $g(n) = O(f(n))$ forem verificadas
- Θ exprime o fato de que as duas funções possuem a **mesma ordem de grandeza assintótica**
- Útil para exprimir limites superiores justos



Notação Θ (cont.)

- Por exemplo, se $f = n^2 - 1$, $g = n^2$ e $h = n^3$, então f é $O(g)$, f é $O(h)$, g é $O(f)$, mas h não é $O(f)$
- Consequentemente, $f = \Theta(g)$, mas f não é $\Theta(h)$

Notação O x Ω x Θ



Notação Θ: Mostre que

- Mostre que $\dfrac{1}{2}n^2 - 3n = \Theta(n^2)$.
- **Resposta:** $c_1 n^2 \leq \dfrac{1}{2}n^2 - 3n \leq c_2 n^2$

  para todo $n \geq n_0$. Dividindo por n^2 temos
- $c_1 \leq \dfrac{1}{2} - \dfrac{3}{n} \leq c_2$
- A desigualdade do lado direito se mantém para qualquer valor $n \geq 1$ escolhendo $c_2 \geq 1/2$
- De maneira similar, a desigualdade do lado esquerdo se mantém para qualquer valor de $n \geq 7$ escolhendo $c_1 \leq 1/14$
- Certamente, outras escolhas de constantes existem, mas é importante é que **ALGUMA** escolha exista

Notação Θ: Mostre que (cont.)

- Mostre que $\frac{1}{2}n^2 - 3n = \Theta(n^2)$.
- **Resposta:** Segue que $\Theta(g(n)) = \{f(n)$: existem constantes positiva c_1, c_2 e n_0, tal que $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para todo $n \geq n_0\}$. Seja $g(n) = n^2$ e $f(n) = 1/2n^2 - 3n$, para $c_1 = 1/14$, $c_2 = 1/2$ e $n_0 = 7$, temos que $1/14n^2 \leq 1/2n^2 - 3n \leq 1/2n^2$ é verdade $\forall n \geq n_0$.

Notação Θ: Intuitivamente

- Intuitivamente, termos de baixa ordem de uma função assintótica positiva podem ser ignorados
- Uma pequena fração de um termo de alta ordem é suficiente para os termos de baixa ordem
- Assim, ajustar c_1 para um valor um pouco menor que o coeficiente do termo de mais alta ordem
- E ajustar o valor de c_2 para um valor um pouco maior permite que a desigualdade da definição da notação Θ seja satisfeita

Problemas Intratáveis

- Suponha que uma máquina moderna pode realizar cerca de 10^{10} operações por segundo
- Suponha uma entrada $n = 100$ para um algoritmo de complexidade 2^n
    - São aproximadamente 10^{30} operações
- Leva-se 10^{20} segundos
- Um dia tem 10^5 segundos
    - Isso significa 10^{15} dias. Ou aproximadamente 10^{13} anos
- A idade do universo foi calculada em cerca de 10^{10} anos …

Complexidade Algoritmos de Ordenação

Name	Best	Average	Worst	Memory	Stable	Method	Other notes
Quicksort	$n \log n$ variation is n	$n \log n$	n^2	$\log n$ on average, worst case is n; Sedgewick variation is $\log n$ worst case	typical in-place sort is not stable; stable versions exist	Partitioning	Quicksort is usually done in place with O(log *n*) stack space.[2][3]
Merge sort	$n \log n$	$n \log n$	$n \log n$	n	Yes	Merging	Highly parallelizable (up to O(log *n*) using the Three Hungarians' Algorithm[4] or, more practically, Cole's parallel merge sort) for processing large amounts of data.
In-place merge sort	—	—	$n \log^2 n$	1	Yes	Merging	Can be implemented as a stable sort based on stable in-place merging.[5]
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No	Selection	
Insertion sort	n	n^2	n^2	1	Yes	Insertion	O(*n* + *d*), in the worst case over sequences that have *d* inversions.
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	No	Partitioning & Selection	Used in several STL implementations.
Selection sort	n^2	n^2	n^2	1	No	Selection	Stable with O(n) extra space, for example using lists.[6]
Timsort	n	$n \log n$	$n \log n$	n	Yes	Insertion & Merging	Makes *n* comparisons when the data is already sorted or reverse sorted.
Cubesort	n	$n \log n$	$n \log n$	n	Yes	Insertion	Makes *n* comparisons when the data is already sorted or reverse sorted.
Shell sort	n	$n \log^2 n$ or $n^{3/2}$	Depends on gap sequence; best known is $n \log^2 n$	1	No	Insertion	Small code size, no use of call stack, reasonably fast, useful where memory is at a premium such as embedded and older mainframe applications.

Referências

- CORMEN, T. H.[et al]. Algoritmos: teoria e prática. 3ª ed. Rio de Janeiro: Elsevier, 2012.
- SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. Estruturas de dados e seus algoritmos. 3. ed. Rio de Janeiro, RJ: LTC, 2010. 302p.