Análise e Projeto de Algoritmos
Análise Assintótica

Prof. André Luiz Moura

andre_moura@ufg.br

Apresentação adaptada (ver referências)

Análise de Algoritmos

- **Correção do algoritmo**
  - “Este algoritmo resolve meu problema?”

Análise de Algoritmos

- **Correção do algoritmo**
  - “Este algoritmo resolve meu problema?”
    - Se sim, em quanto tempo?

Análise de Algoritmos

- **Correção do algoritmo**
  - "Este algoritmo resolve meu problema?"
    - Se sim, em quanto tempo?

- **Complexidade do algoritmo**
  - "Em quanto tempo esse algoritmo resolve meu problema?

Análise de Algoritmos

- **Correção do algoritmo**
  - “Este algoritmo resolve meu problema?”
    - Se sim, em quanto tempo?

- **Complexidade do algoritmo**
  - “Em quanto tempo esse algoritmo resolve meu problema?

- **Memória requerida**
  - “Quanta memória esse algoritmo requer?”

Análise de Algoritmos

- **Correção do algoritmo**
  - "Este algoritmo resolve meu problema?"
    - Se sim, em quanto tempo?
- **Complexidade do algoritmo** → **Ordem de Grandeza**
  - "Em quanto tempo esse algoritmo resolve meu problema?
- **Memória requerida**
  - "Quanta memória esse algoritmo requer?"

Análise de Algoritmos

- **Correção do algoritmo**
  - “Este algoritmo resolve meu problema?”
    - Se sim, em quanto tempo?

De uma função

↑

- **Complexidade do algoritmo** → Ordem de Grandeza
  - “Em quanto tempo esse algoritmo resolve meu problema?
- **Memória requerida**
  - “Quanta memória esse algoritmo requer?”

Análise de Algoritmos

Que determine quantas operações esse algoritmo executará com base na entrada.

↑

De uma função

↑

- **Correção do algoritmo**
  - “Este algoritmo resolve meu problema?”
    - Se sim, em quanto tempo?
- **Complexidade do algoritmo** → **Ordem de Grandeza**
  - “Em quanto tempo esse algoritmo resolve meu problema?
- **Memória requerida**
  - “Quanta memória esse algoritmo requer?”

Análise de Algoritmos

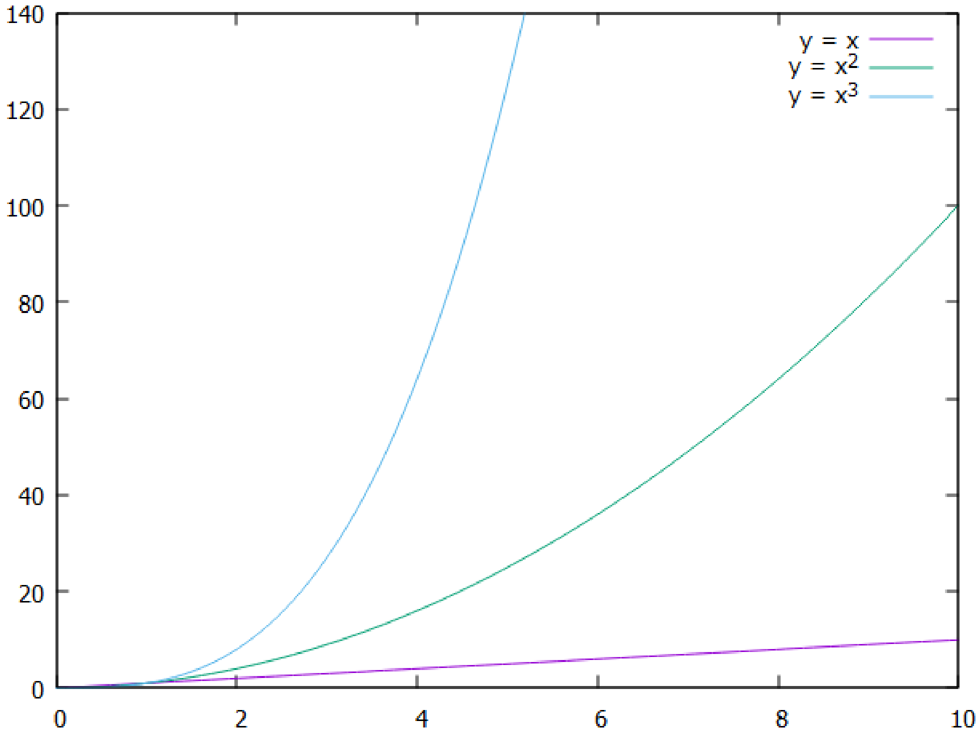
- **Correção do algoritmo**
  - "Este algoritmo resolve meu problema?"
    - Se sim, em quanto tempo?

Quantas operações esse algoritmo executará?

- **Complexidade do algoritmo** → **Ordem de Grandeza**
  - "Em quanto tempo esse algoritmo resolve meu problema?
- **Memória requerida**
  - "Quanta memória esse algoritmo requer?"

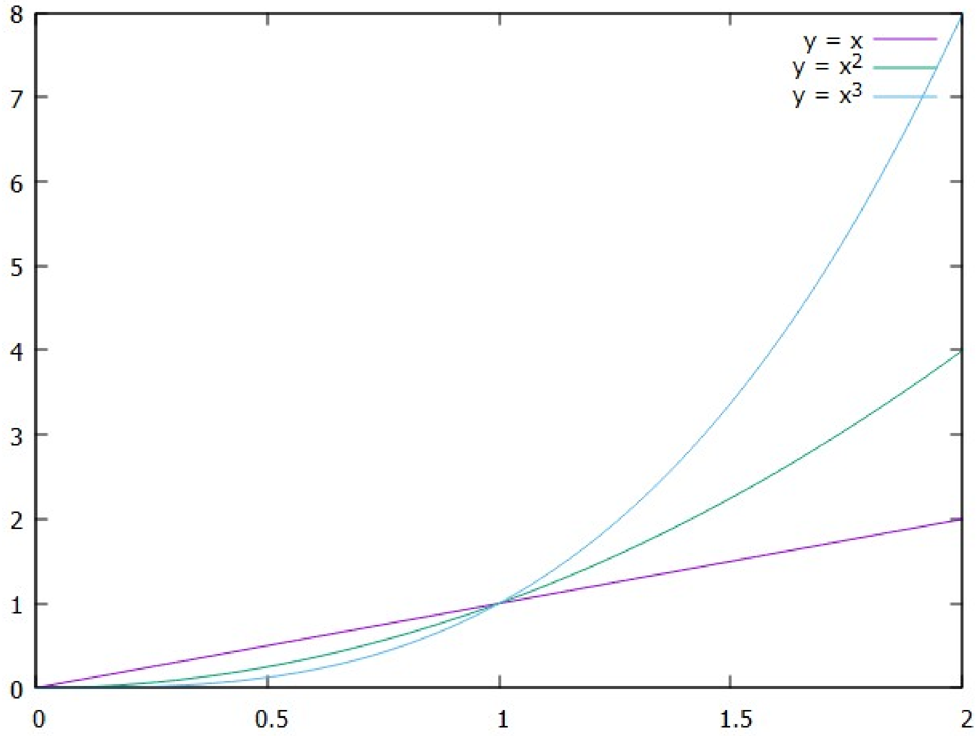
Análise de Algoritmos

- **Ordem de grandeza de uma função**
  - É a taxa/velocidade de crescimento de uma função.
  - Uma função pode crescer devagar, rápido ou muito rápido.



Análise de Algoritmos

- **Ordem de grandeza de uma função**
  - É a taxa/velocidade de crescimento de uma função.
  - Uma função pode crescer devagar, rápido ou muito rápido.



Notação Assintótica

- É uma abstração que nos permite estudar a taxa de crescimento de funções. Assim, é possível, por exemplo, focar no que ocorre com **f(*n*)** quando ***n*** cresce indefinidamente.
- São formalismos que indicam limitantes (mínimo e máximo) para funções e que escondem alguns detalhes.
- Ela é necessária porque, para valores pequenos de ***n***, os comportamentos das funções podem variar.

Notação Assintótica

- A notação é representada pela ordem de grandeza, que é obtida a partir da expressão de função. Isso se é feito em 3 passos:

  Passo 1: Manter apenas o termo de maior ordem da expressão.

  Passo 2: Remover a constante multiplicativa associada ao termo de maior ordem.

  Passo 3: Incluir o termo obtido no Passo 2 e colocá-lo como parâmetro de uma função assintótica.

- **Por exemplo:** $f(n) = 2n^3 + 4000n^2 + 10000n – 47$
  - Após aplicação do Passo 1: $2n^3$
  - Após aplicação do Passo 2: $\mathbf{n^3}$

Notação Assintótica

- A notação é representada pela ordem de grandeza, que é obtida a partir da expressão de função. Isso se é feito em 3 passos:

  Passo 1: Manter apenas o termo de maior ordem da expressão.
  Passo 2: Remover a constante multiplicativa associada ao termo de maior ordem.
  Passo 3: Incluir o termo obtido no Passo 2 e colocá-lo como parâmetro de uma função assintótica.

Função assintótica: Uma função que descreve o comportamento de uma função **f(n)** à medida que o tamanho de **n** cresce, tendendo ao infinito.

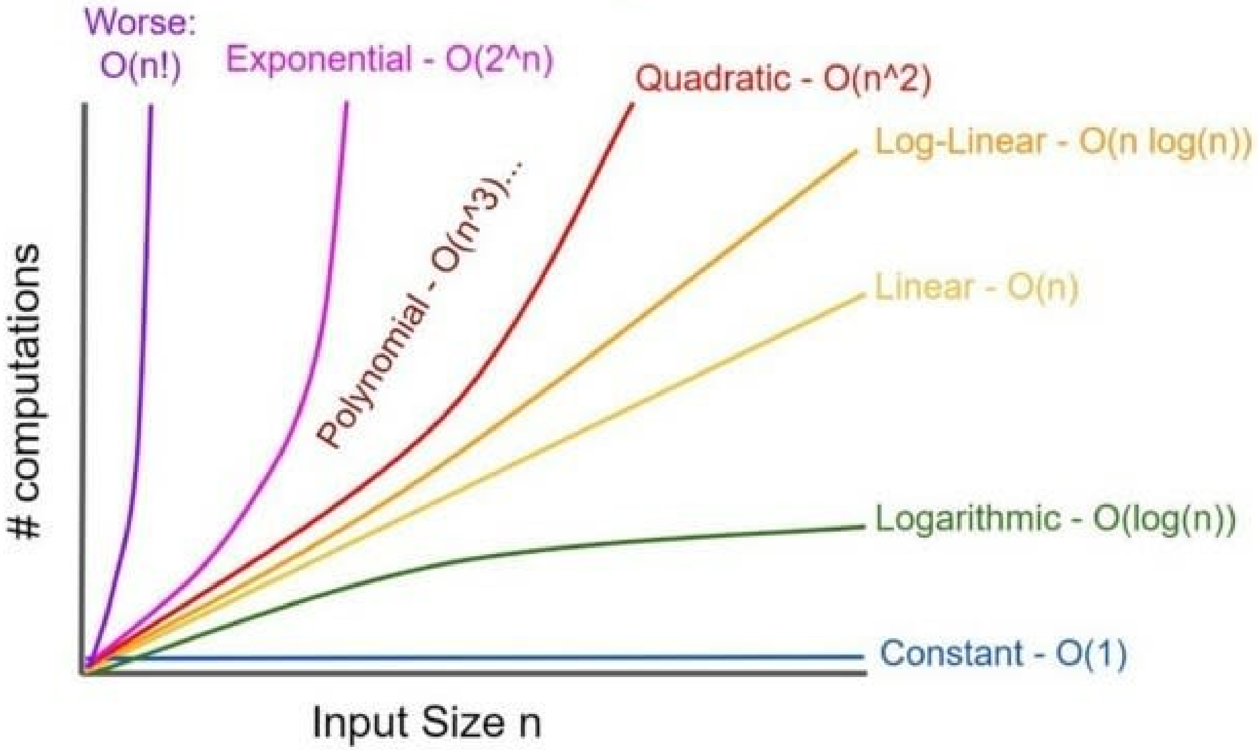
- **Por exemplo:** $f(n) = 2n^3 + 4000n^2 + 10000n – 47$
  - Após aplicação do Passo 1: $2n^3$
  - Após aplicação do Passo 2: $\mathbf{n^3}$

Notações O, Ω e Θ

- As três notações fornecem limitantes.
- Por exemplo, dada uma função ***f*(*n*)**, ela pode ser maior que alguma coisa, menor que alguma coisa ou possuir as duas características ao mesmo tempo.
- As notações assintóticas indicam limitantes para as funções a partir de determinados valores de ***n***.
- A notação O fornece o limitante superior, $f(n) \leq c.g(n)$.
- A notação Ω fornece o limitante inferior, $f(n) \geq c.g(n)$.
- A notação Θ fornece, simultaneamente, os limitantes inferior e superior, $c1.g1(n) \leq f(n) \leq c2.g2(n)$.

Notação O

- É uma notação assintótica usada para analisar a eficiência de um algoritmo à medida que os valores de entrada crescem, considerando sempre o pior cenário.



Notação O – Complexidades mais comuns

O(1) = Complexidade constante o tempo de execução do algoritmo independe do tamanho da entrada é bem rápido.

O(log(n)) = Complexidade logarítmica o tempo de execução pode ser considerado menor do que uma constante grande. É super rápido

O(n) = Complexidade linear o algoritmo realiza um número fixo de operações sobre cada elemento da entrada

O(n log(n)) = Típico de algoritmos que dividem um problema em subproblemas, resolve cada subproblema de forma independente, e depois combina os resultados

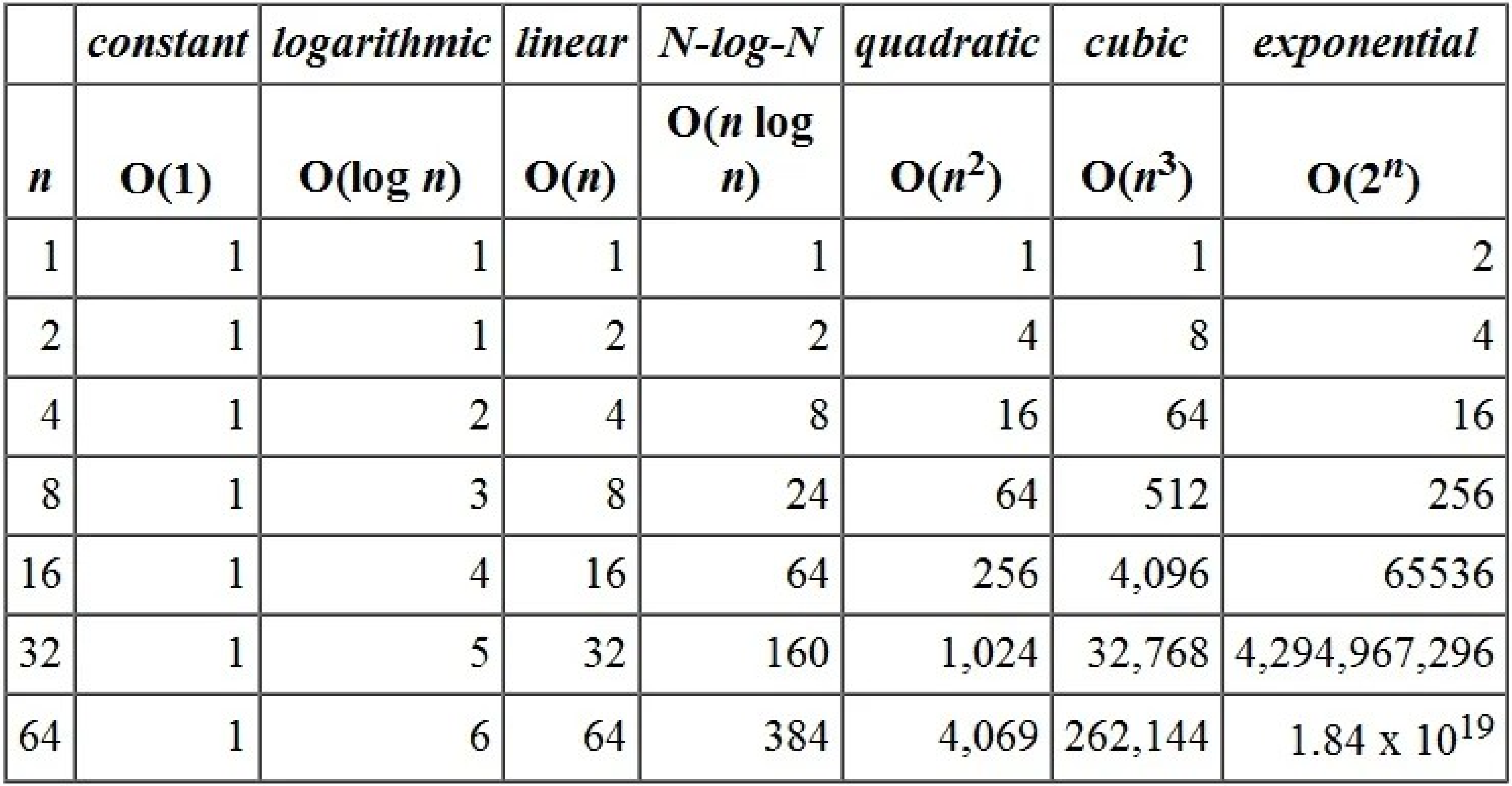
O(n^2) = Complexidade quadrática. Típico de algoritmos que operam sobre pares dos elementos de entrada

O(n^3) = Complexidade cúbica que é útil para resolver problemas pequenos como multiplicação de matrizes

O(2^n) = Complexidade exponencial. Típicos de algoritmos que fazem busca exaustiva (força bruta) para resolver um problema

O(n!) = Complexidade fatorial. É normalmente encontrado ao analisar a complexidade de algoritmos de força bruta, que tentam todas as possibilidades para problemas de otimização combinatória

Notação O – Comparativo das complexidades



	constant	*logarithmic*	*linear*	*N-log-N*	*quadratic*	*cubic*	*exponential*
n	**O(1)**	**O(log n)**	**O(n)**	**O(n log n)**	**O(n^2)**	**O(n^3)**	**O(2^n)**
1	1	1	1	1	1	1	2
2	1	1	2	2	4	8	4
4	1	2	4	8	16	64	16
8	1	3	8	24	64	512	256
16	1	4	16	64	256	4,096	65536
32	1	5	32	160	1,024	32,768	4,294,967,296
64	1	6	64	384	4,069	262,144	1.84×10^{19}

Notação O

- A notação **O** limita funções superiormente, $f(n) \leq c.g(n)$.
- Seja *n* um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que

  $f(n) = O(g(n))$ [ou $f(n)$ é $O(g(n))$]

  se existem constantes positivas **c** e $\mathbf{n_o}$, tais que

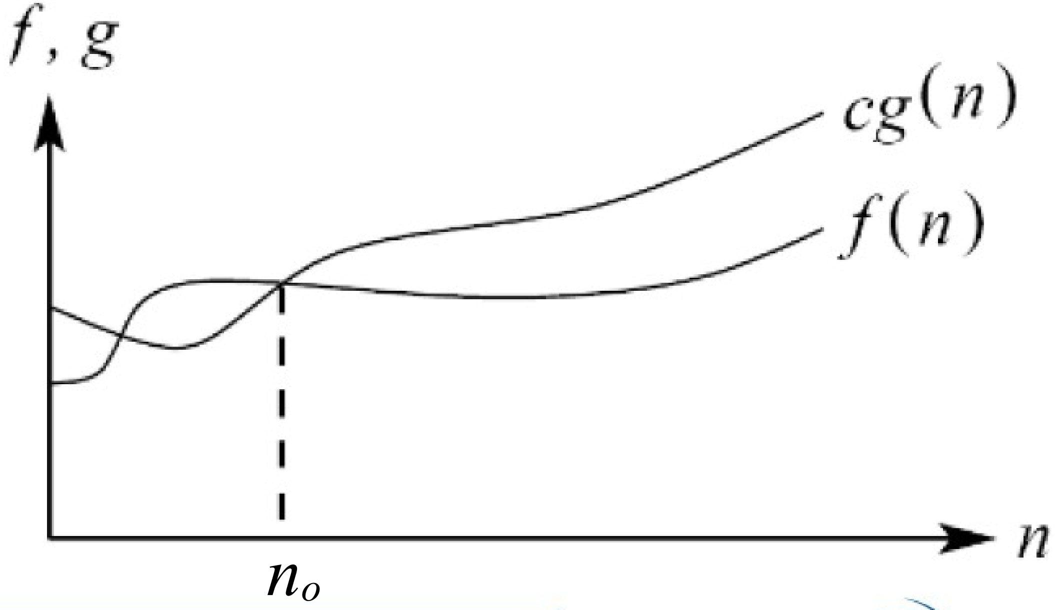
  $f(n) \leq c.g(n) \quad \forall \quad n \geq n_o$. (Leia-se: f(n) é limitada a c.g(n).)

- É importante destacar que deve existir **algum valor de *n*** em que c.g(n) esteja acima de f(n).

Notação O

- Seja *n* um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o** (n_o > 0).
- Exemplos: **f(n) = 5n + 3**

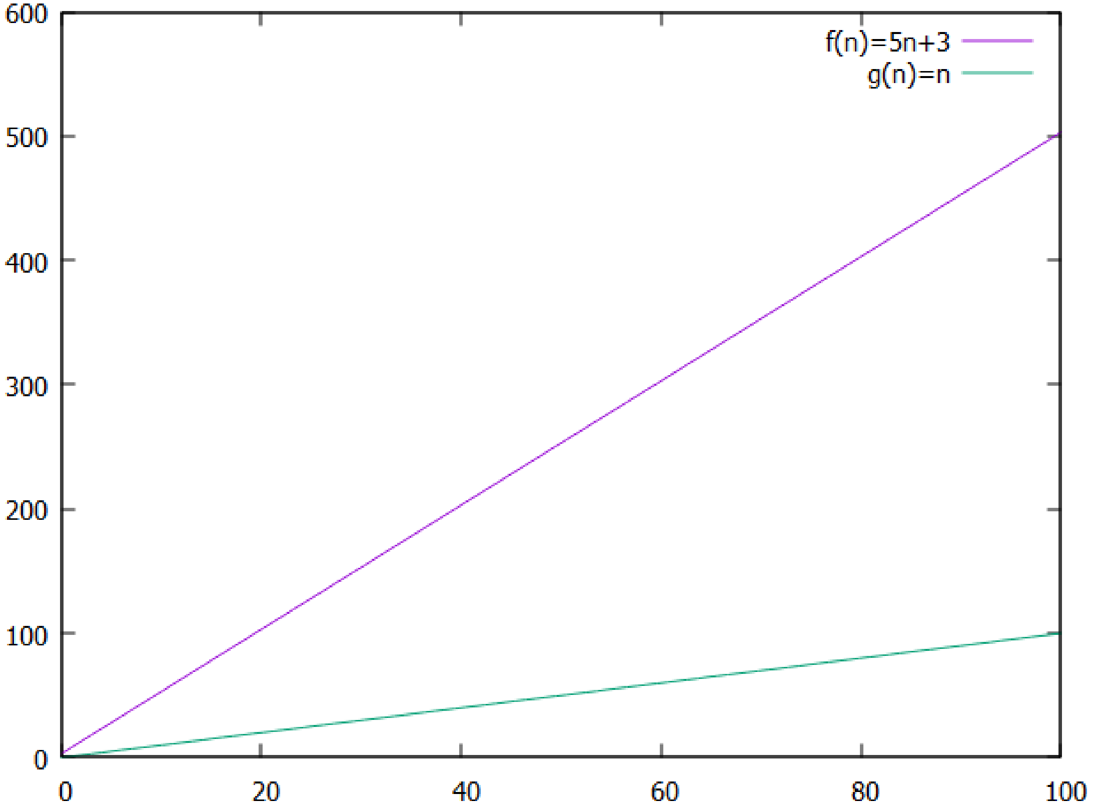
  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)



Notação O

- Demonstração de que f(n) <= c.g(n), ou seja, que 5n + 3 = O(n)

  Gráfico para f(n) = 5n + 3 e g(n) = n



Notação O

- Demonstração de que f(n) <= c.g(n), ou seja, que 5n + 3 = O(n)

$5n + 3 <= c.n$

1. Dividir ambos os lados por n a fim de isolar o valor da constante c.

$$\frac{5n + 3}{n} <= \frac{c.n}{n}$$

$$5 + \frac{3}{n} <= c$$

2. Encontrar um valor inicial para n_o, tal que $n >= n_o$.

Quando n >= 1, o termo $\frac{3}{n}$ tende a zero à medida que n aumenta.

Assim, escolhendo o valor no = 1, temos:

$$5 + \frac{3}{1} <= c$$

Resolvendo a inequação, temos:

$5 + 3 <= c$

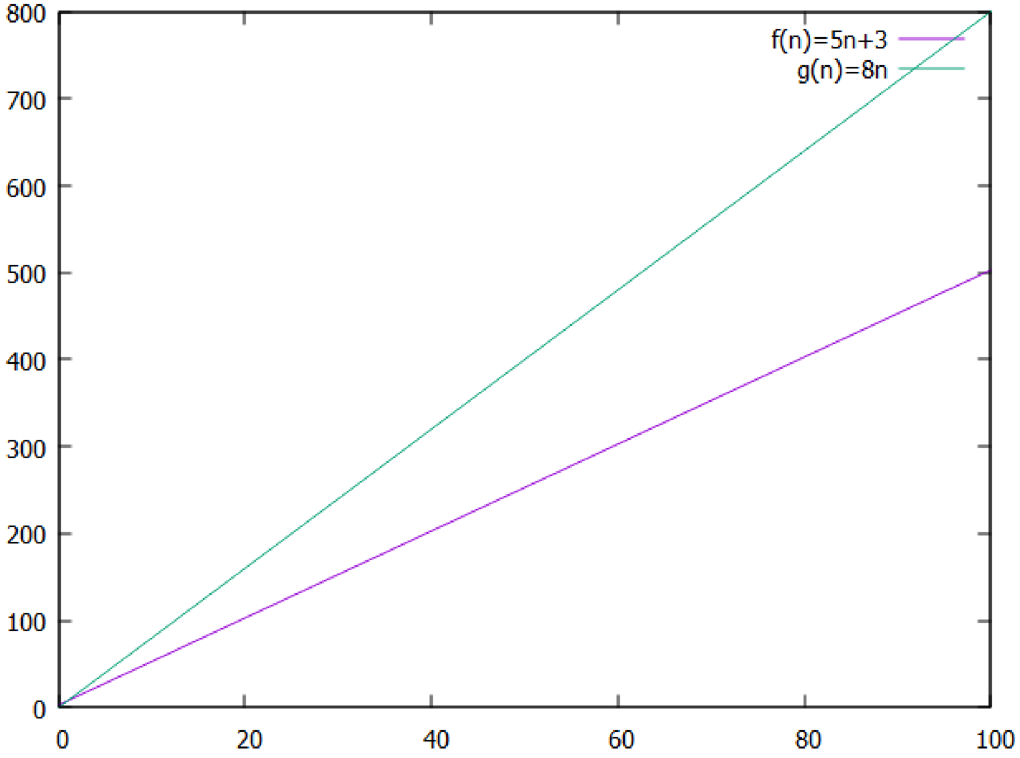
$8 <= c$

Notação O

- Demonstração de que f(n) <= c.g(n), ou seja, que 5n + 3 = O(n)

Portanto, pode-se afirmar que **5n + 3 = O(n)**, (f(n) <= c.g(n), para $n_o = 1$ e c = 8).
Isso significa que **5n + 3 <= 8n**.

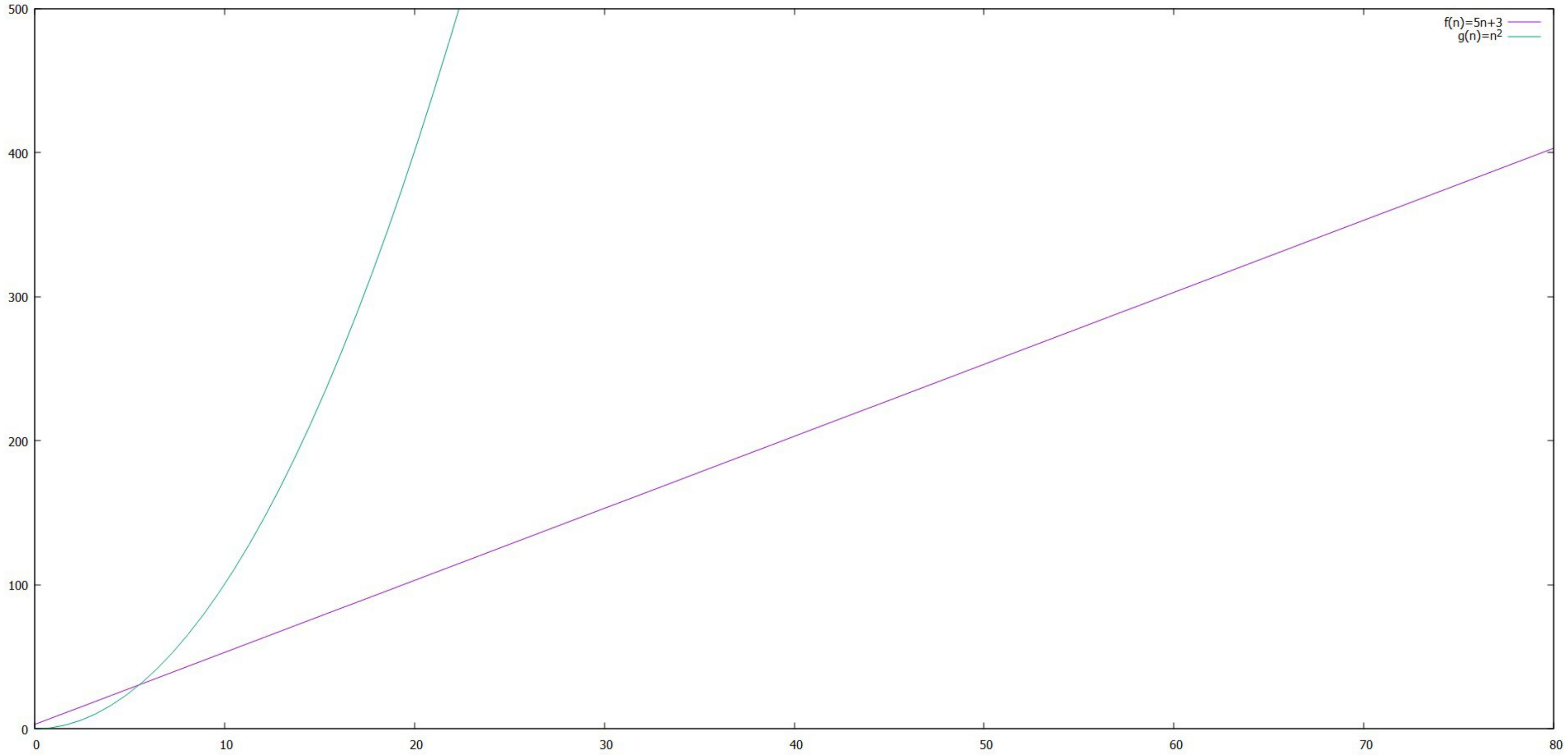
Gráfico para **f(n) = 5n + 3, g(n) = 8n**



Notação O

- 1ª demonstração de que f(n) <= c.g(n), ou seja, que $5n + 3 = O(n^2)$

Gráfico para $\mathbf{f(n) = 5n + 3, g(n) = n^2}$



Notação O

- 1ª demonstração de que f(n) <= c.g(n), ou seja, que $5n + 3 = O(n^2)$

$5n + 3 <= c.n^2$

1. Dividir ambos os lados por **n^2** a fim de isolar o valor da constante **c**.

$$\frac{5n + 3}{n^2} <= \frac{c.n^2}{n^2}$$

$$\frac{5}{n} + \frac{3}{n^2} <= c$$

2. Encontrar um valor inicial para n_o, tal que $n >= n_o$.

Quando n >= 1, o termo $\frac{3}{n^2}$ tende a zero à medida que n aumenta.

Assim, escolhendo o valor no = 1, temos:

$$\frac{5}{1} + \frac{3}{1^2} <= c$$

Resolvendo a inequação, temos:

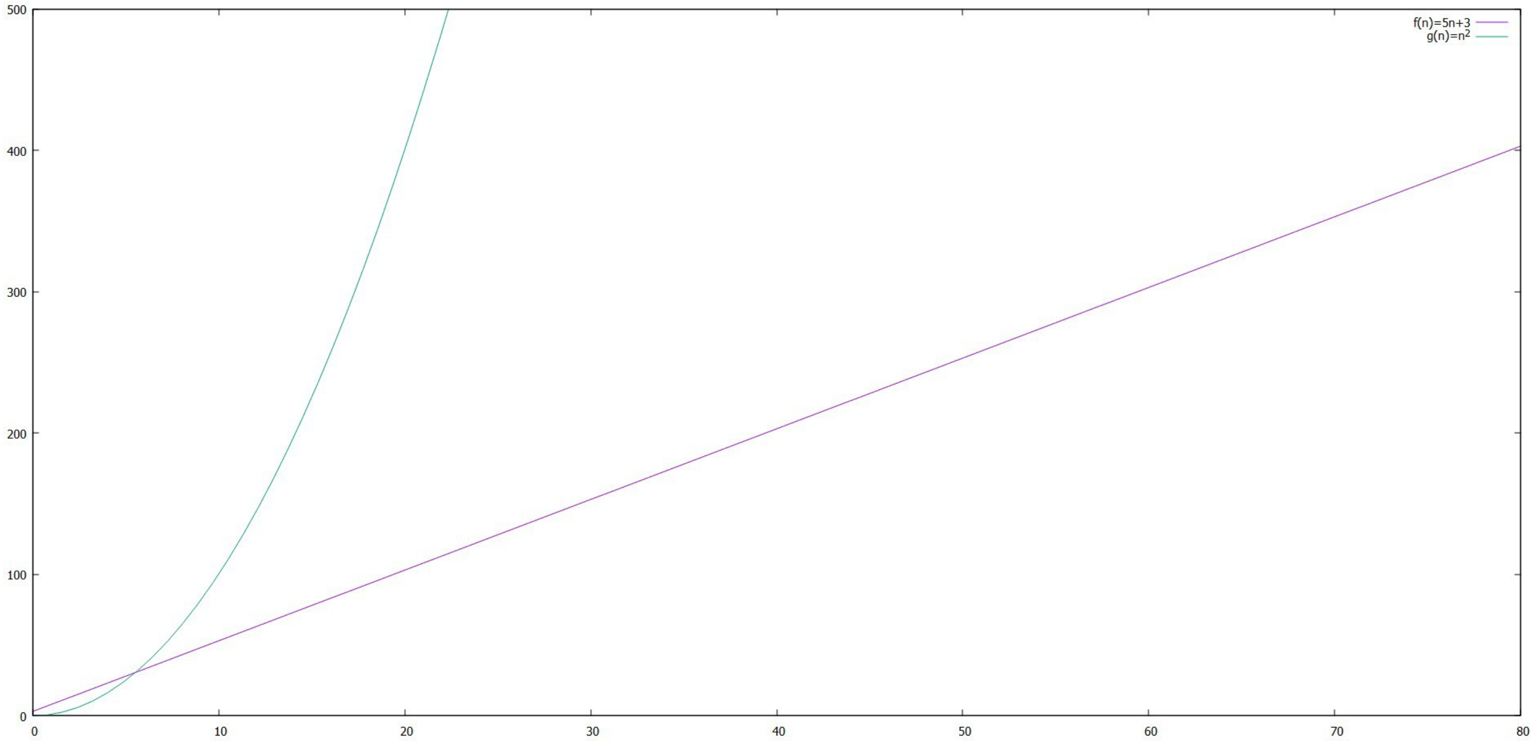
5 + 3 <= c

8 <= c

Notação O

- 1ª demonstração de que f(n) <= c.g(n), ou seja, que 5n + 3 = O(n^2)

Portanto, pode-se afirmar que 5n + 3 = O(n^2), ou seja, f(n) é O(g2(n), para n >= 1 e c=8.
Isso significa que 5n + 3 <= 8n^2.
Gráfico para **f(n) = 5n + 3, g(n) = 8n^2**



Notação O

- 2ª demonstração de que $f(n) <= c.g(n)$, ou seja, que $5n + 3 = O(n^2)$

$5n + 3 <= c.n^2$

1. Dividir ambos os lados por **c** a fim de isolar o valor do termo **n²**.

$$\frac{5n + 3}{c} <= \frac{c.n^2}{c}$$

$$\frac{5n + 3}{c} <= n^2$$

2. Encontrar um valor inicial para c, tal que c >= 1.

Assim, escolhendo o valor c = 1, temos:

$$\frac{5n + 3}{1} <= n^2 => 5n + 3 <= n^2$$

Passar o termo dominante n^2 para o lado esquerdo, a fim de, posteriormente, obter as raízes (valores de n) por meio de uma equação de 2º Grau, usando a fórmula de Bhaskara.

$-n^2 + 5n + 3 <= 0$ (-1)

$n^2 - 5n - 3 >= 0$

Notação O

- 2ª demonstração de que f(n) <= c.g(n), ou seja, que $5n + 3 = O(n^2)$

Resolver essa desigualdade quadrática, encontrando os pontos em que a parábola $n^2 - 5n - 3$ cruza o eixo x (ou seja, onde a desigualdade é satisfeita). Para fazer isso, usar a fórmula quadrática:

$$n = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},$$

onde a, b e c são os coeficientes na equação $n^2 - 5n - 3 = 0$ (Equação de 2º Grau).

Nesse caso:

a= 1, b= −5, c = −3
Aplicando a fórmula quadrática:
$D = b^2 – 4ac$ (D = Delta = Discriminante)
D = (-5)^2 – (4 (1) (−3))
D = 25 + 12
D = 37 (Delta > 0 indica que há dois valores para x, onde x = n.)

$$x = \frac{-b+-\text{sqrt}(D)}{2a}$$

Notação O

- 2ª demonstração de que f(n) <= c.g(n), ou seja, que 5n + 3 = O(n^2)

a= 1, b= −5, c = −3

$$x = \frac{-b+-\text{sqrt}(D)}{2a}$$

$$x = \frac{-(-5) +- \text{sqrt}(37)}{2(1)}$$

sqrt(37) = 6,0827625

$$x1 = \frac{5 + 6{,}0827625}{2}$$

$$x1 = \frac{11{,}0827625}{2}$$

x1 = Teto(5,54138125) => x1 = 6

$$x2 = \frac{5 – 6{,}0827625}{2}$$

x2 = -1,0827625 / 2

x2 = -0,5413825 (Valor inválido, pois n >= n0, n0 >= 1)

Notação O

- 2ª demonstração de que f(n) <= c.g(n), ou seja, que 5n + 3 = O(n^2)

Agora, deve-se verificar se os valor de n satisfaz a inequação 5n + 3 <= n^2

Para n = 6 e c = 1, temos:
5(6) + 3 <= $(6)^2$
30 + 3 <= 36
33 <= 36

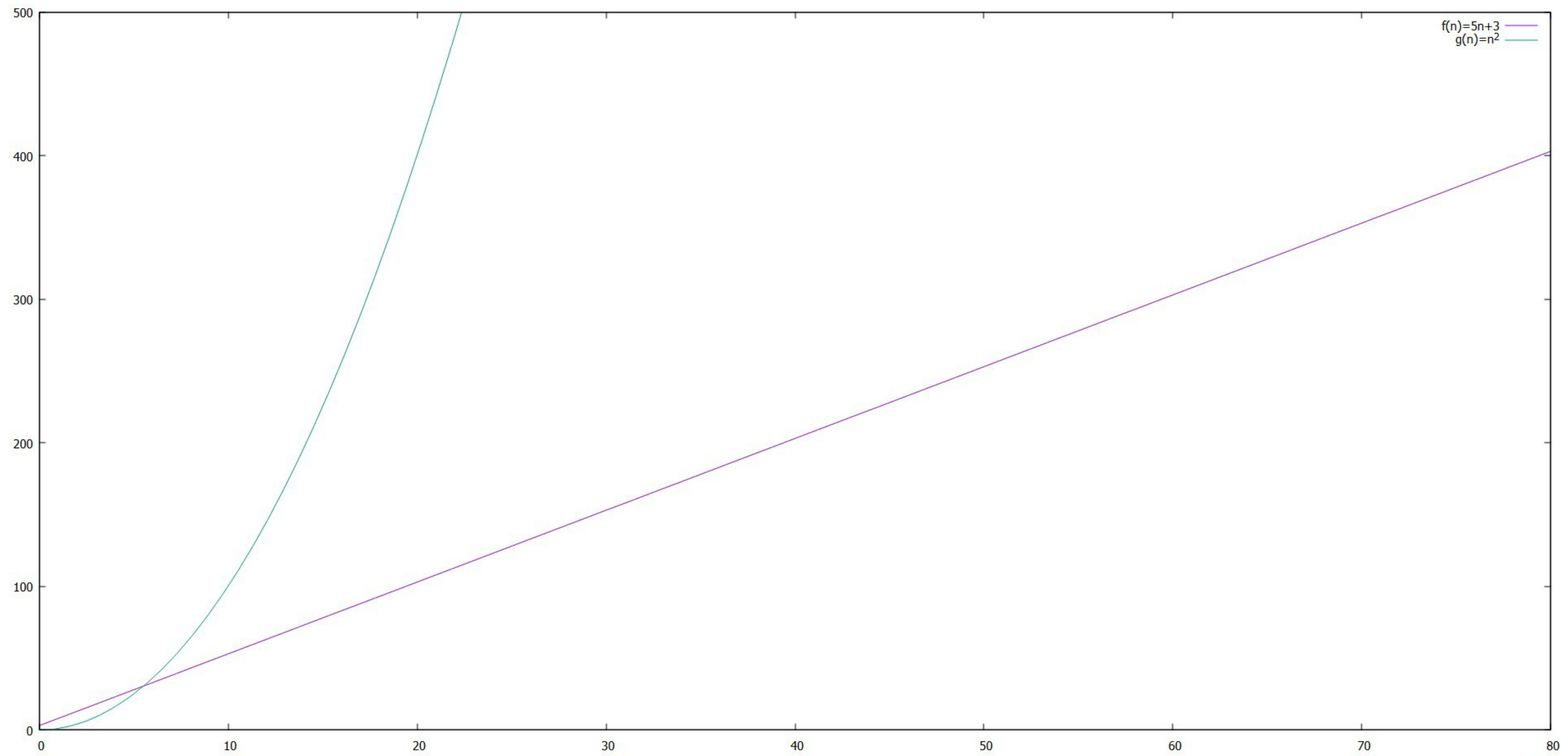
Portanto, sendo 5n + 3 = O(n^2), conclui-se que **f(n) = O(g2(n)), para n >= 6 e c=1**.

Como f(n) é O(n^2), podemos também afirmar que:
f(n) é O(n^3), f(n) é O(n^{10}), f(n) é O(n log n).

Notação O

- Demonstrar que f(n) <= c.g(n), ou seja, que 5n + 3 = O(n^2)

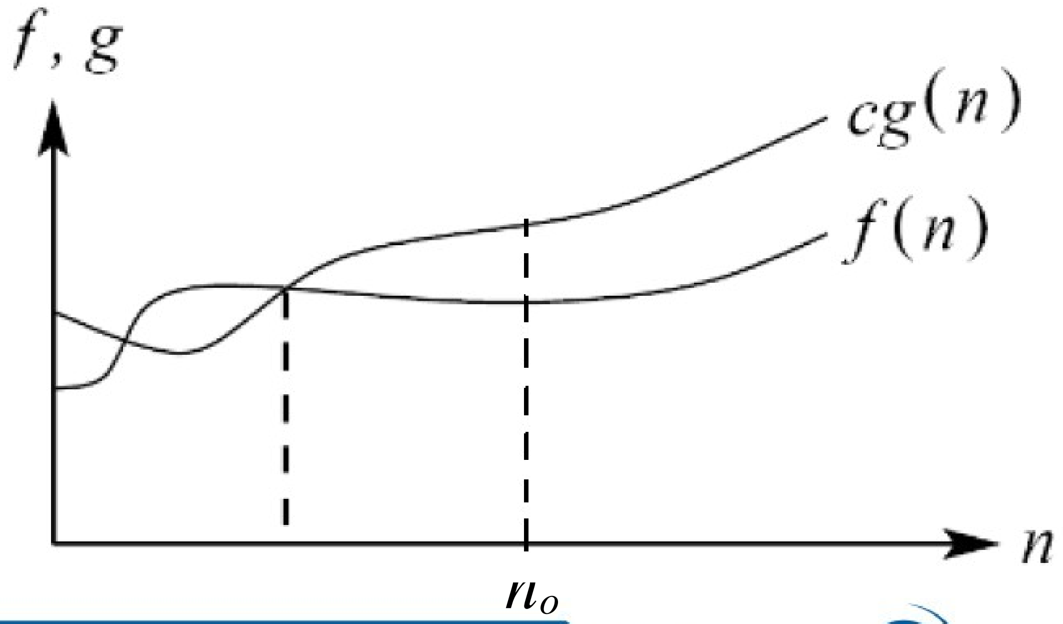
Gráfico para **$f(n) = 5n + 3$, $g1(n) = n^2$**



Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)



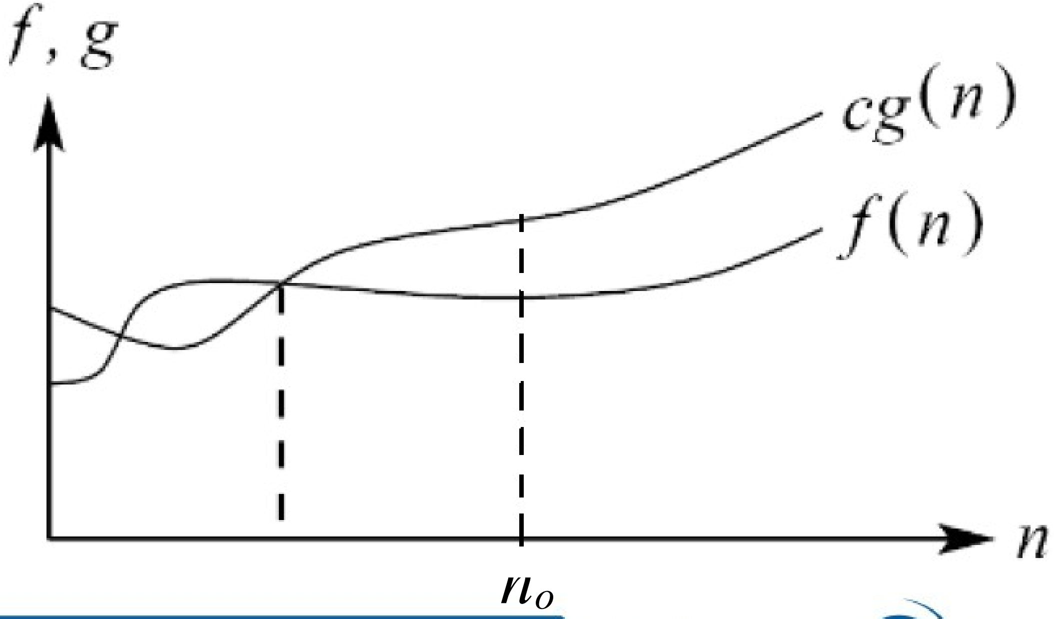
Será que f(n) é ≤ sqrt(n)?

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)

  = $n^{1/2}$



Será que f(n) é ≤ sqrt(n)?

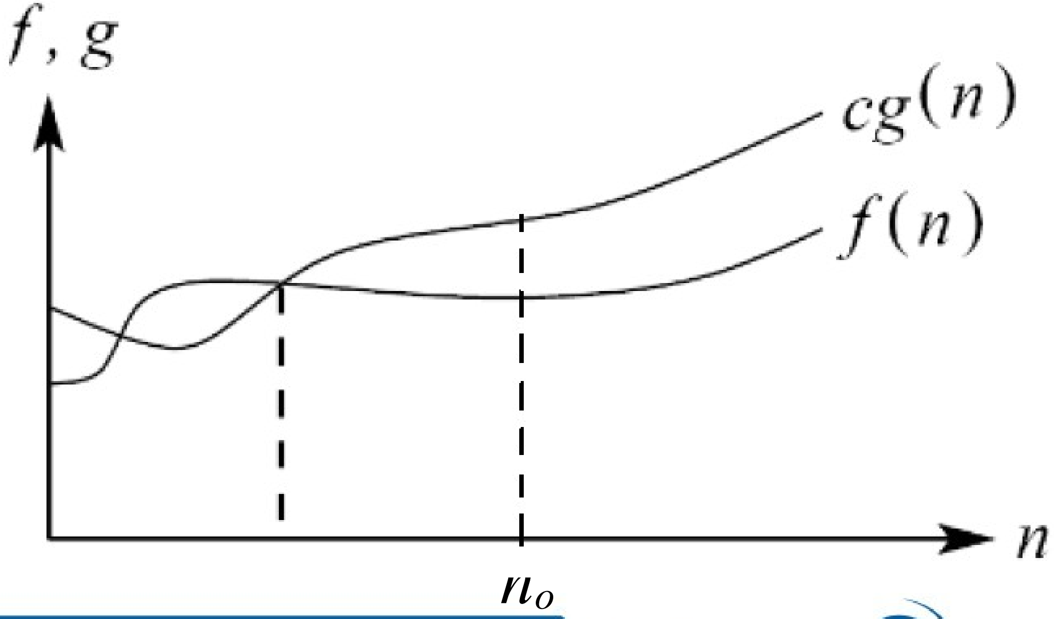
Intuitivamente, não, pois, f(n) > sqrt(n).

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem constantes positivas c e n$_o$, tais que f(n) ≤ c.g(n) ∀ n ≥ n$_o$.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)

  = $n^{1/2}$



Demonstração:

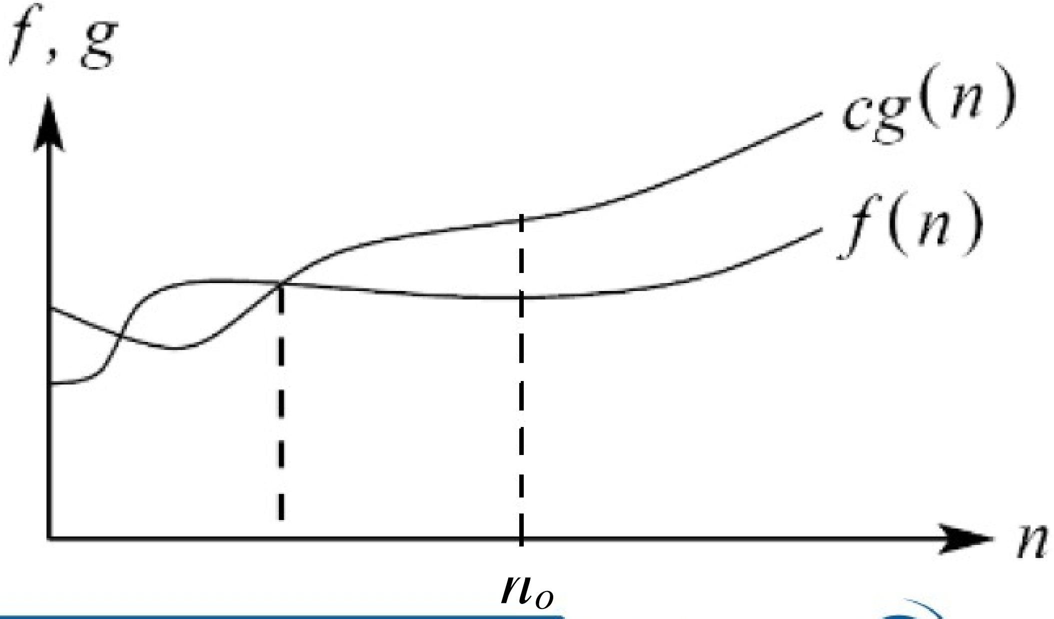
5n + 3 ≤ c.sqrt(n)?

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem constantes positivas c e n_o, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)

  = $n^{1/2}$



Demonstração:

5n + 3 ≤ c.sqrt(n)?

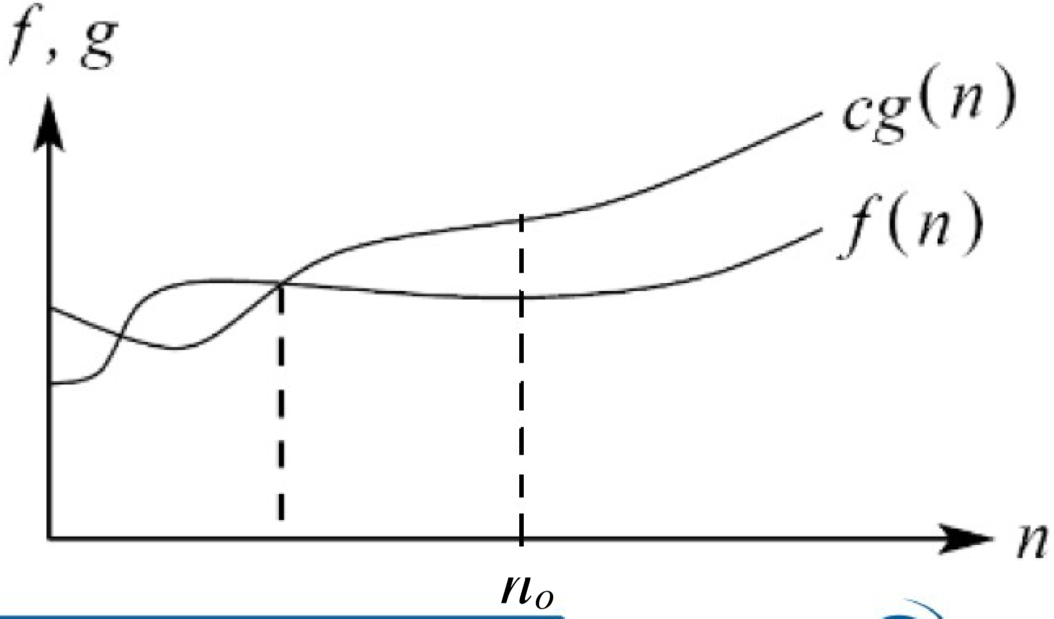
$$\frac{5n + 3}{\text{sqrt}(n)} \leq \frac{c.\text{sqrt}(n)}{\text{sqrt}(n)}$$

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem** **constantes positivas c e n$_o$**, **tais que f(n) ≤ c.g(n) ∀ n ≥ n$_o$.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)

  = $n^{1/2}$



Demonstração:

5n + 3 ≤ c.sqrt(n)?

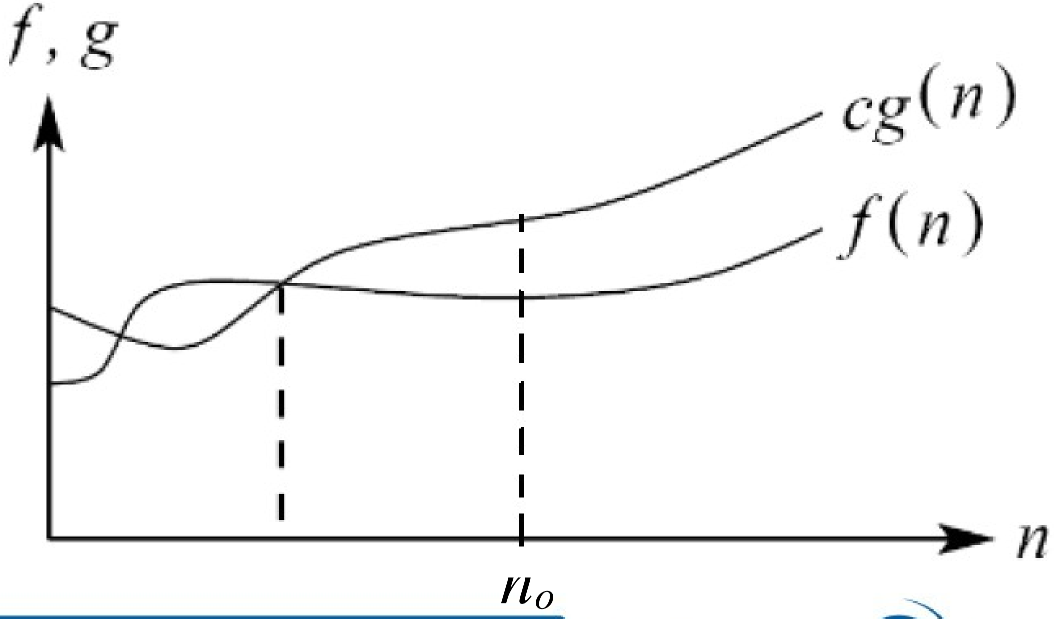
$$\frac{5n + 3}{\text{sqrt}(n)} \leq c$$

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)

  = $n^{1/2}$



Demonstração:

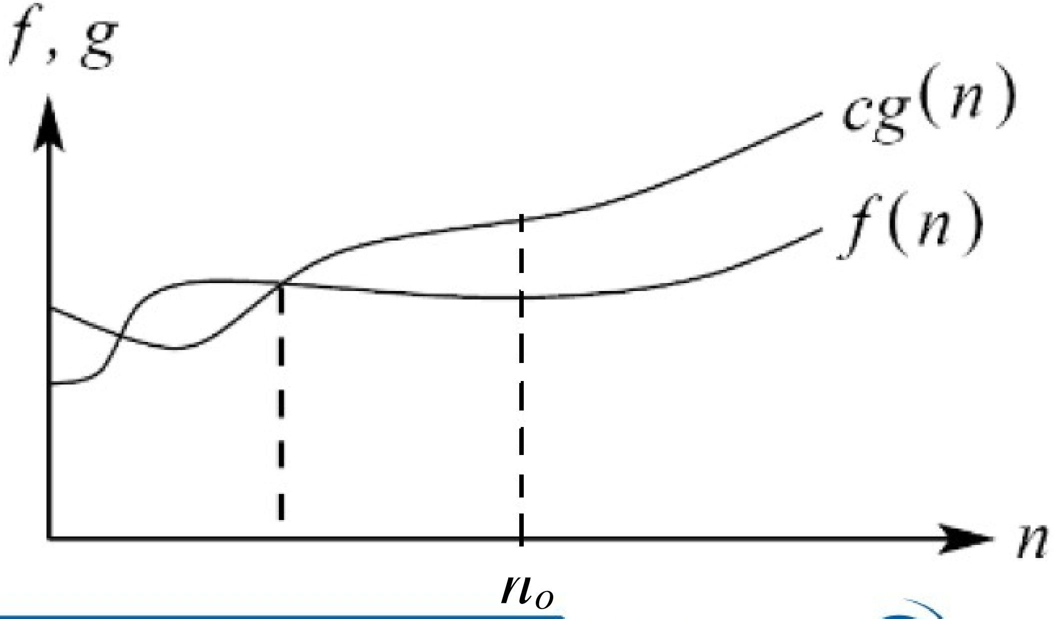
5n + 3 ≤ c.sqrt(n)?

$$\frac{5n}{\text{sqrt(n)}} + \frac{3}{\text{sqrt(n)}} \leq c$$

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem constantes positivas c e n_o, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)
  = $n^{1/2}$



Demonstração:

5n + 3 ≤ c.sqrt(n)?

$$5*\text{sqrt}(n) + \frac{3}{\text{sqrt}(n)} \leq c$$

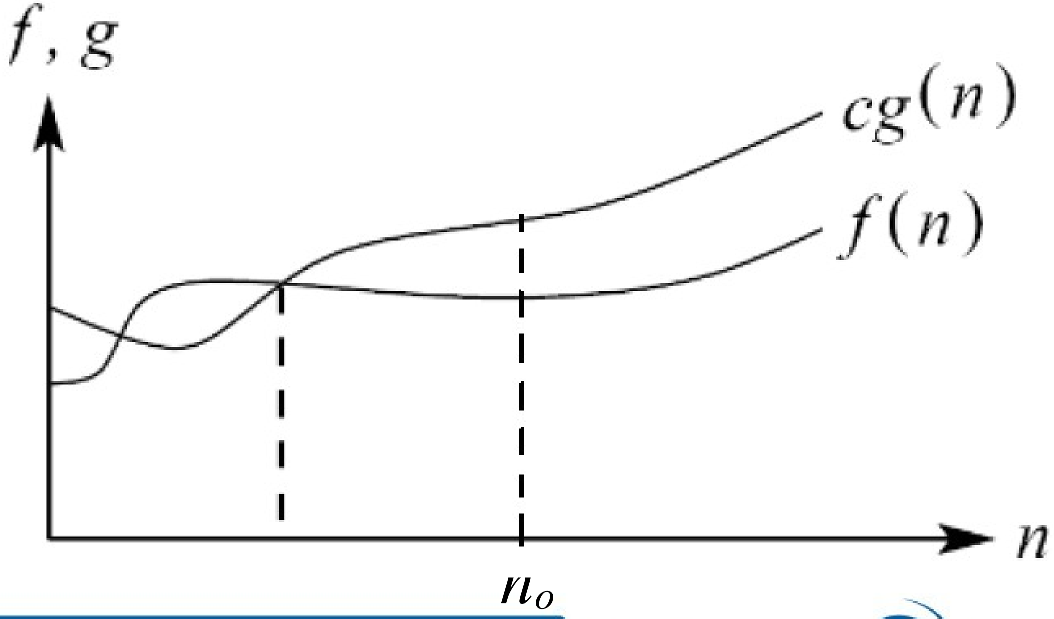
c é uma constante, que independe de n

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem constantes positivas c e n_o, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)

  $= n^{1/2}$



Demonstração:

5n + 3 ≤ c.sqrt(n)?

$$5*\text{sqrt}(n) + \overset{0}{\cancel{\frac{3}{\text{sqrt}(n)}}} \le c$$

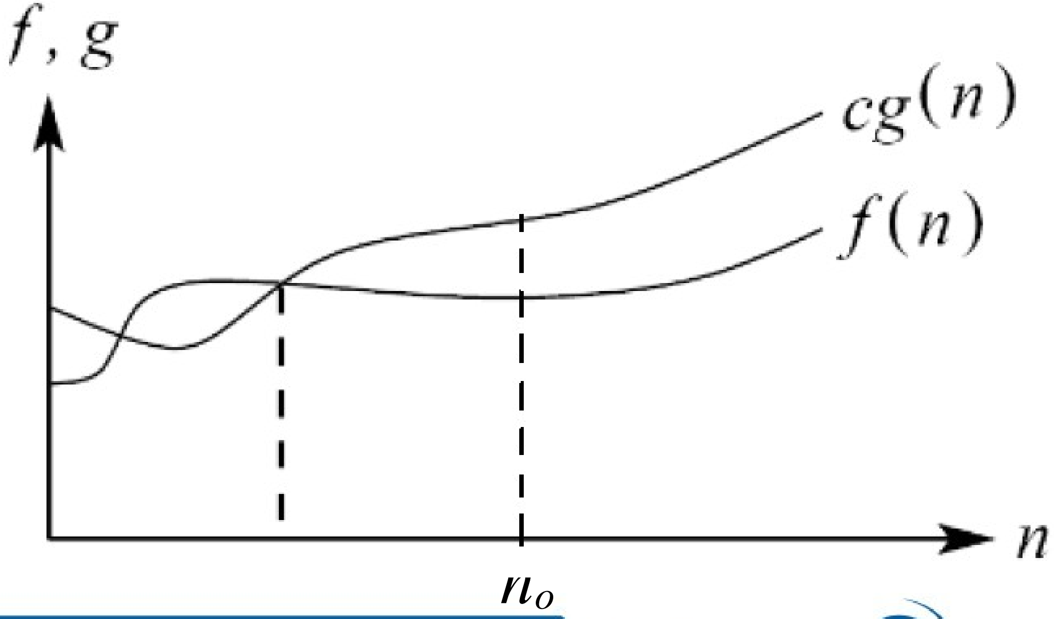
c é uma constante, que independe de n

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem constantes positivas c e n_o, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)

= $n^{1/2}$



Demonstração:

5n + 3 ≤ c.sqrt(n)?

5*sqrt(n) ≤ c

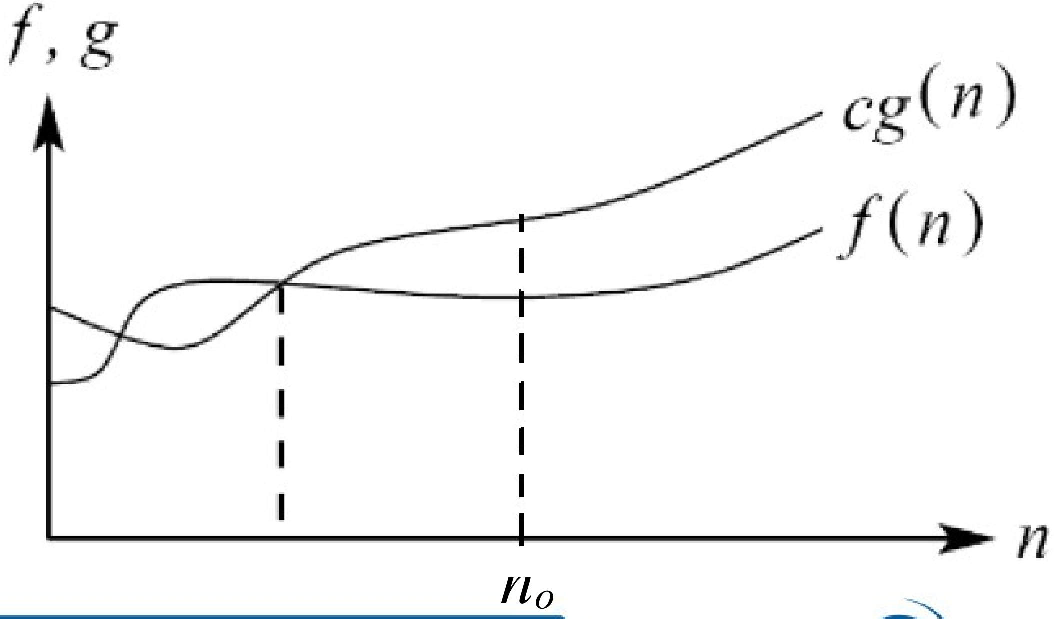
A expressão à esquerda da desigualdade tende se tornar enorme à medida que **n** cresce.

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem <u>constantes positivas c e n$_o$</u>, tais que f(n) ≤ c.g(n) ∀ n ≥ n$_o$.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)

  = $n^{1/2}$



Demonstração:

5n + 3 ≤ c.sqrt(n)?

5*sqrt(n) ≤ c

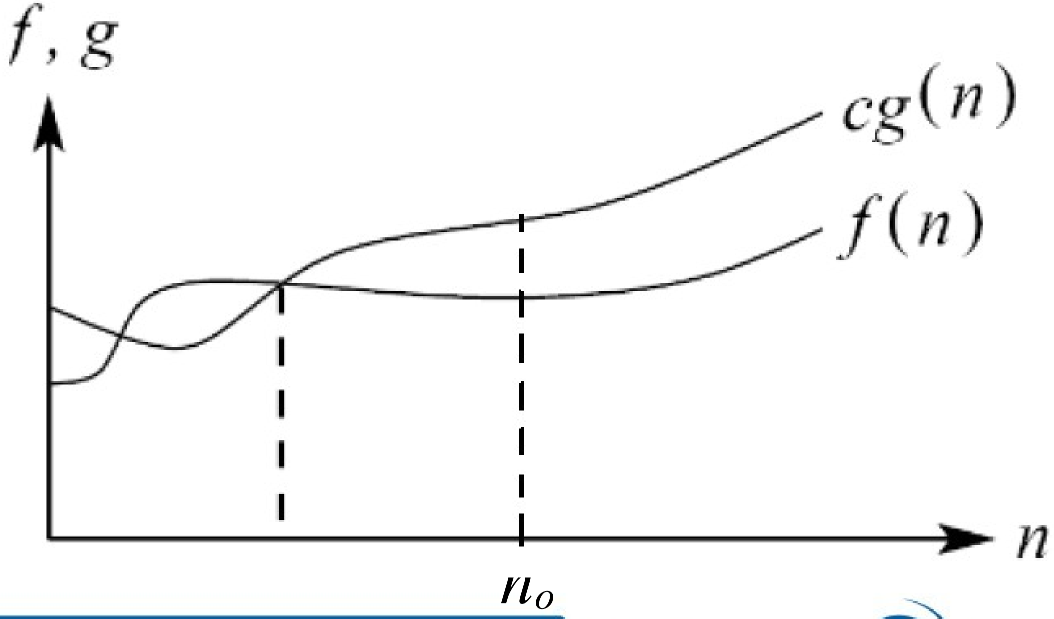
Mesmo que c seja 1 trilhão, em algum momento 5*sqrt(n) > c. Logo, f(n) não é O(g3(n)).

Notação O

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = O(g(n)) se existem constantes positivas c e n_o, tais que f(n) ≤ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)

  = $n^{1/2}$



Demonstração:

5n + 3 ≤ c.sqrt(n)?

5*sqrt(n) ≤ c

Logo, f(n) não é O(g3(n)). Ou seja, **f(n) não é O(sqrt(n))**. Assim, f(n) também não é O(log(n)) e f(n) não é O(1).

Notação Ω

- A notação **Ω** limita funções inferiormente, $f(n) \geq c.g(n)$.
- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que

  $f(n) = \Omega(g(n))$ [ou $f(n)$ é $\Omega(g(n))$]

  se existem constantes positivas **c** e **n_o**, tais que

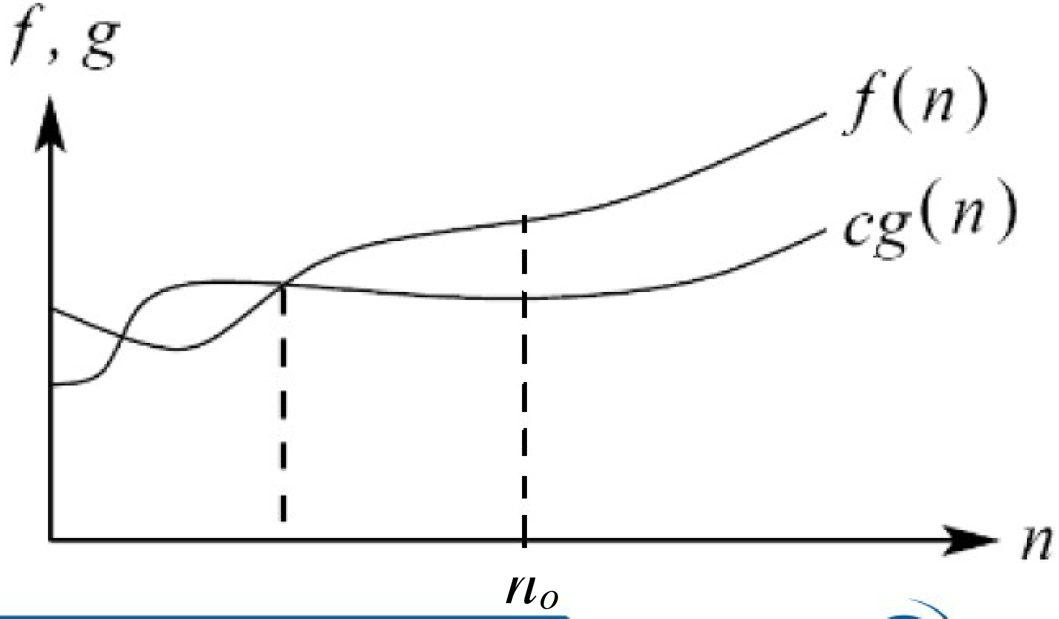
  $f(n) \geq c.g(n) \quad \forall \; n \geq n_o$. (Leia-se: f(n) é limitada a c.g(n))

- É importante destacar que deve existir algum valor de *n* em que *c*.*g*(*n*) esteja abaixo de *f*(*n*).

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)



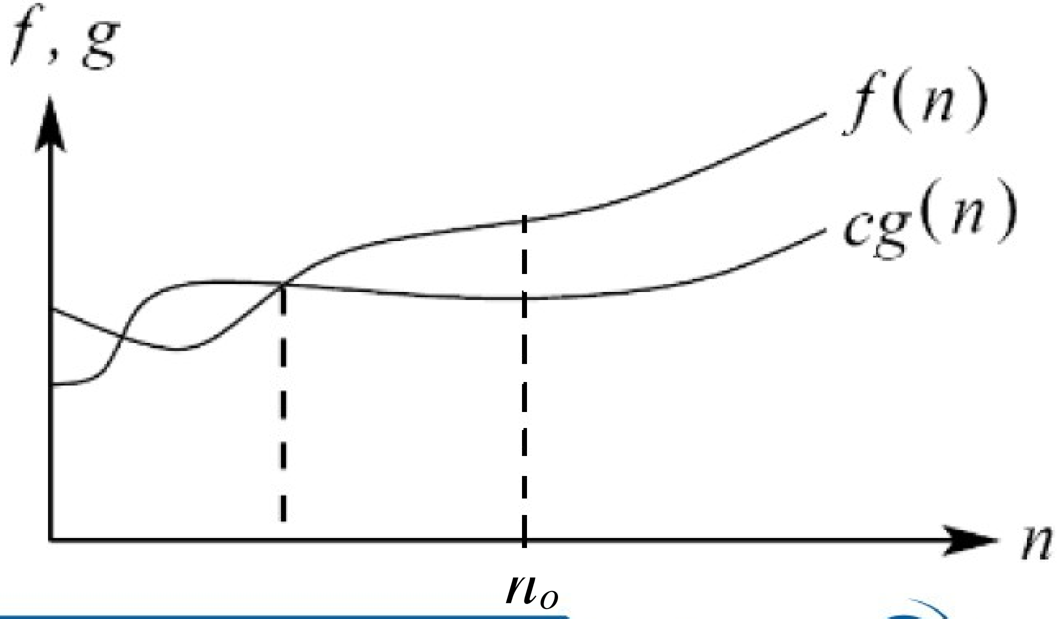
Demonstração de que f(n) ≥ g1(n):

5n + 3 ≥ 5n

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)



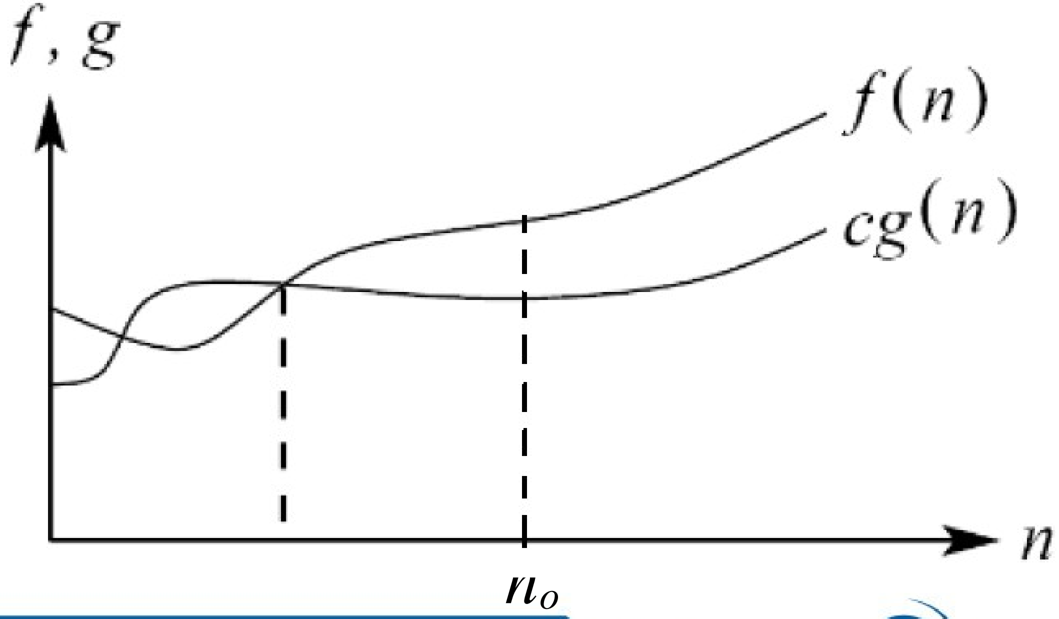
Demonstração de que f(n) ≥ g1(n):

5n + 3 ≥ 5n = c.g1(n)

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que f(n) ≥ g1(n):

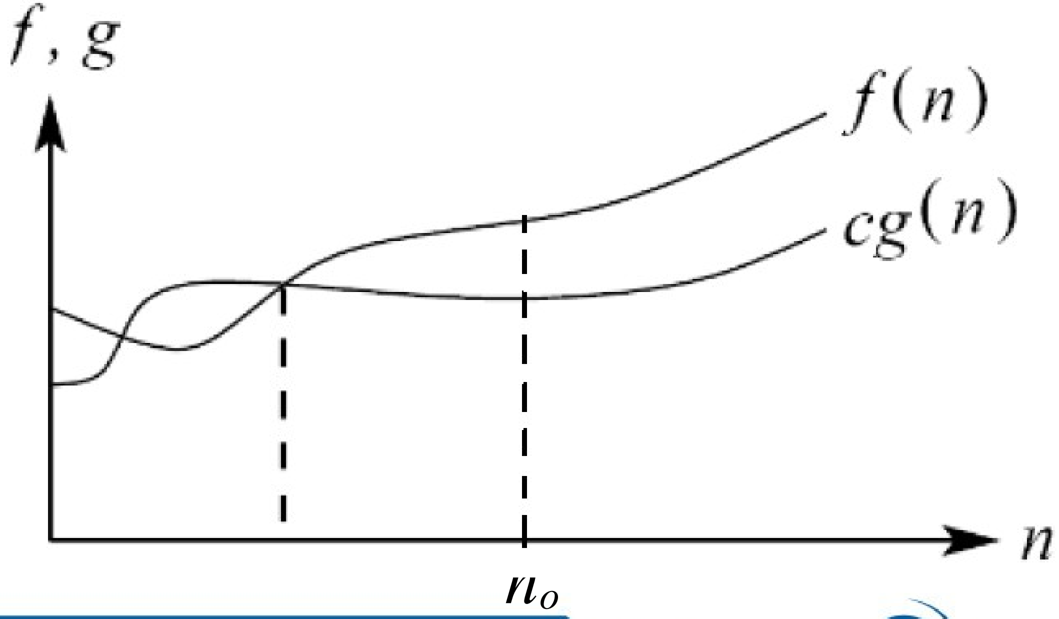
5n + 3 ≥ 5n = c.g1(n)

Logo, f(n) = Ω(g1(n)). Ou seja, **f(n) = Ω(n)**.

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem constantes positivas c e n_o, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que f(n) ≥ g2(n):

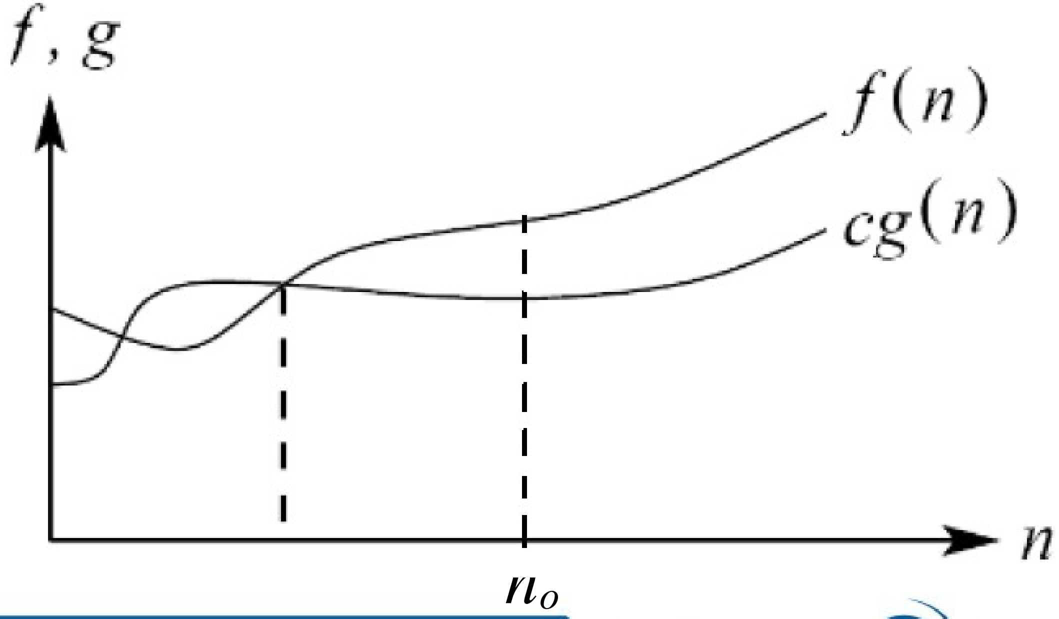
5n + 3 ≥ c.n^2?

Intuitivamente, não!

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **$f(n) = 5n + 3$**

  $g1(n) = n$ $g2(n) = n^2$ $g3(n) = sqrt(n)$



Demonstração de que f(n) ≥ g2(n):

$5n + 3 \geq c.n^2$?

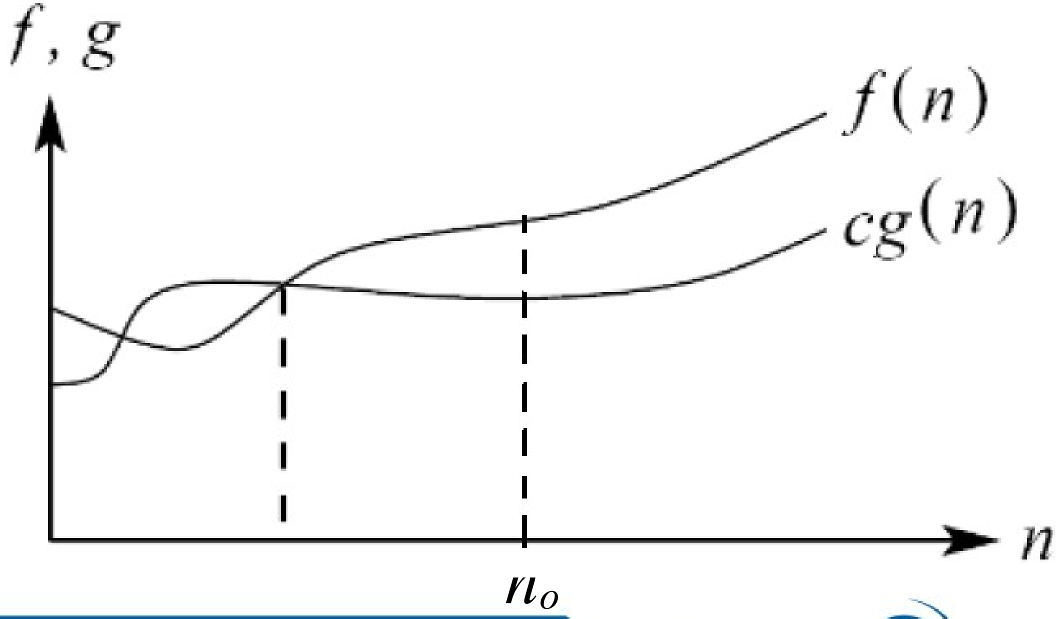
Intuitivamente, não!

Mas, vamos demonstrar por que não.

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que não é verdade que f(n) ≥ g2(n):

$5n + 3 \geq c.n^2?$

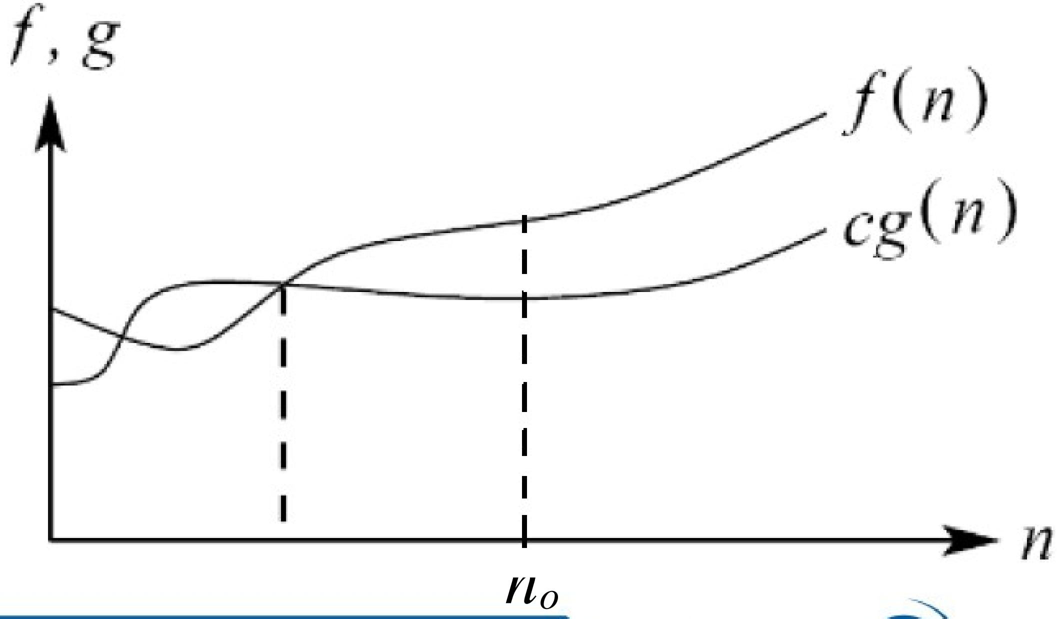
$$\frac{5n + 3}{n^2} \geq c$$

$$\frac{5n +}{n^2} \frac{3}{n^2} \geq c$$

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que não é verdade que f(n) ≥ g2(n):

$5n + 3 \geq c.n^2?$

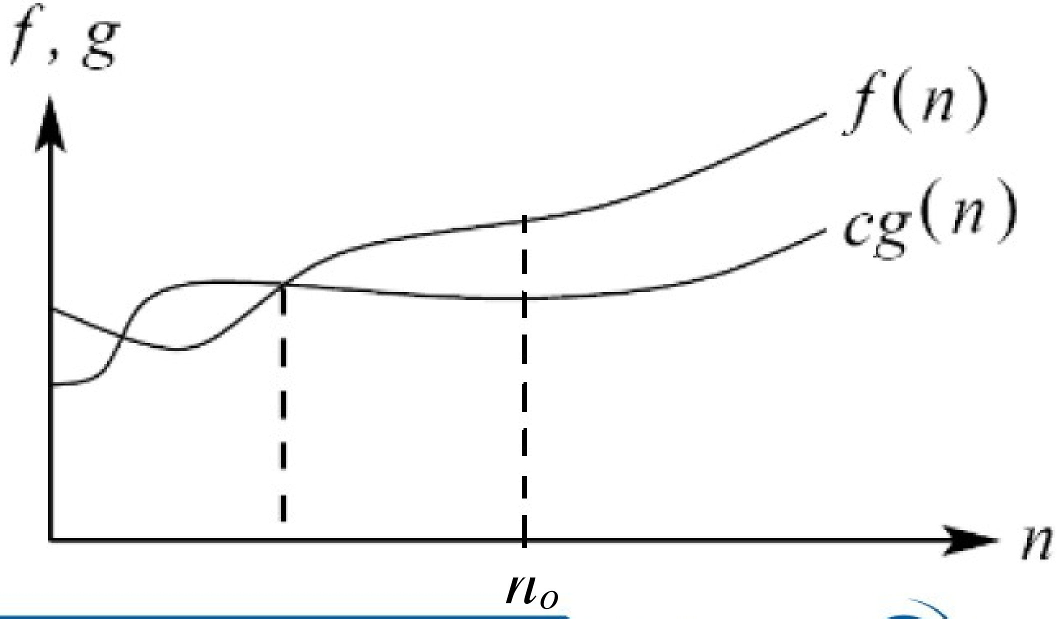
$$\frac{5n + 3}{n^2} \geq c$$

$$\frac{5}{n} + \frac{3}{n^2} \geq c$$

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n₀</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n₀.**
- Exemplos: **$f(n) = 5n + 3$**

  $g1(n) = n$ $g2(n) = n^2$ $g3(n) = sqrt(n)$



Demonstração de que não é verdade que f(n) ≥ g2(n):

$5n + 3 \geq c.n^2?$

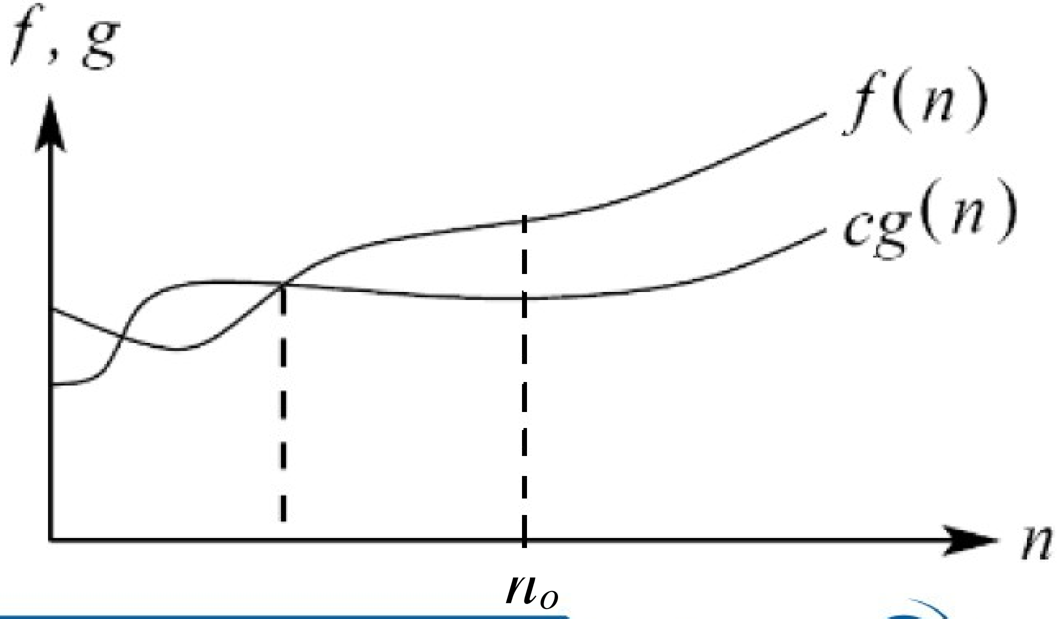
$$\frac{5n + 3}{n^2} \geq c$$

$$\overset{0}{\cancel{\frac{5}{n}}} + \overset{0}{\cancel{\frac{3}{n^2}}} \geq c$$

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que $f(n) \geq c.g(n)\ \forall\ n \geq n_o$.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que não é verdade que f(n) ≥ g2(n):

$5n + 3 \geq c.n^2$?

$$\frac{5n + 3}{n^2} \geq c$$

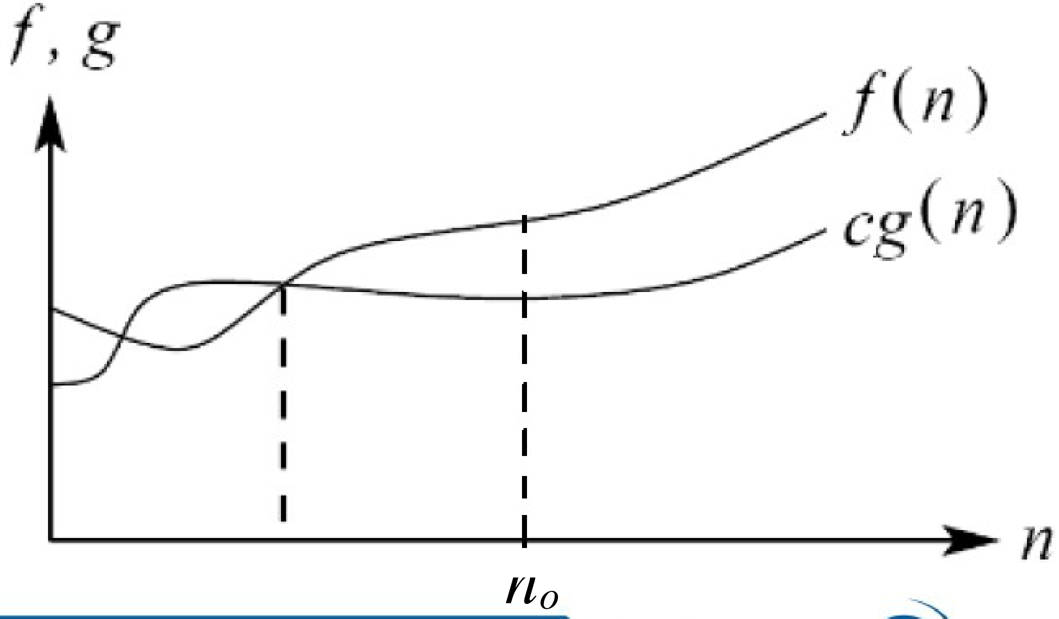
$$\overset{0}{\cancel{\frac{5}{n}}} + \overset{0}{\cancel{\frac{3}{n^2}}} \geq c$$

À medida que n cresce, os dois termos tendem a 0.

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **$f(n) = 5n + 3$**

  $g1(n) = n$ $g2(n) = n^2$ $g3(n) = sqrt(n)$



Demonstração de que não é verdade que f(n) ≥ g2(n):

$5n + 3 \geq c.n^2?$

$$\frac{5n + 3}{n^2} \geq c$$

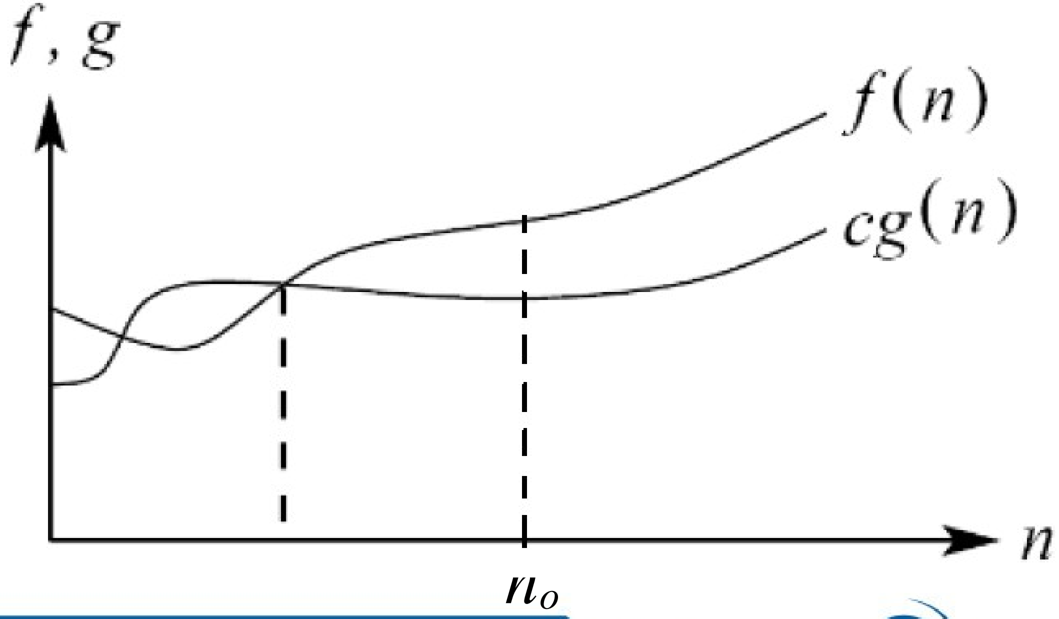
$$\overset{0}{\cancel{\frac{5}{n}}} + \overset{0}{\cancel{\frac{3}{n^2}}} \geq c$$

Como encontrar uma constante positiva menor que o valor da expressão, sendo que a constante c deve ser maior que 0 (zero)?!

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n_o</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que não é verdade que f(n) ≥ g2(n):

$5n + 3 \geq c.n^2?$

$$\frac{5n + 3}{n^2} \geq c$$

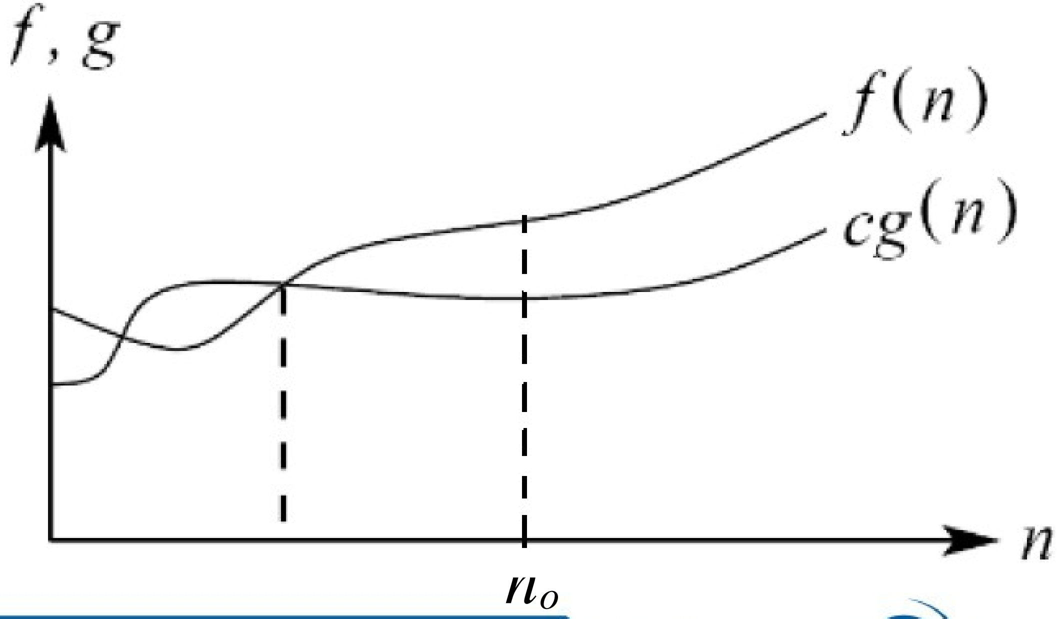
$$\overset{0}{\cancel{\frac{5}{n}}} + \overset{0}{\cancel{\frac{3}{n^2}}} \geq c$$

Logo, f(n) não é Ω(g2(n)). Ou seja, **f(n) não é Ω(n^2).**

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem constantes positivas c e n_o, tais que f(n) ≥ c.g(n) ∀ n ≥ n_o.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que não é verdade que f(n) ≥ g2(n):

5n + 3 ≥ c.n^2?

$$\frac{5n + 3}{n^2} \geq c$$

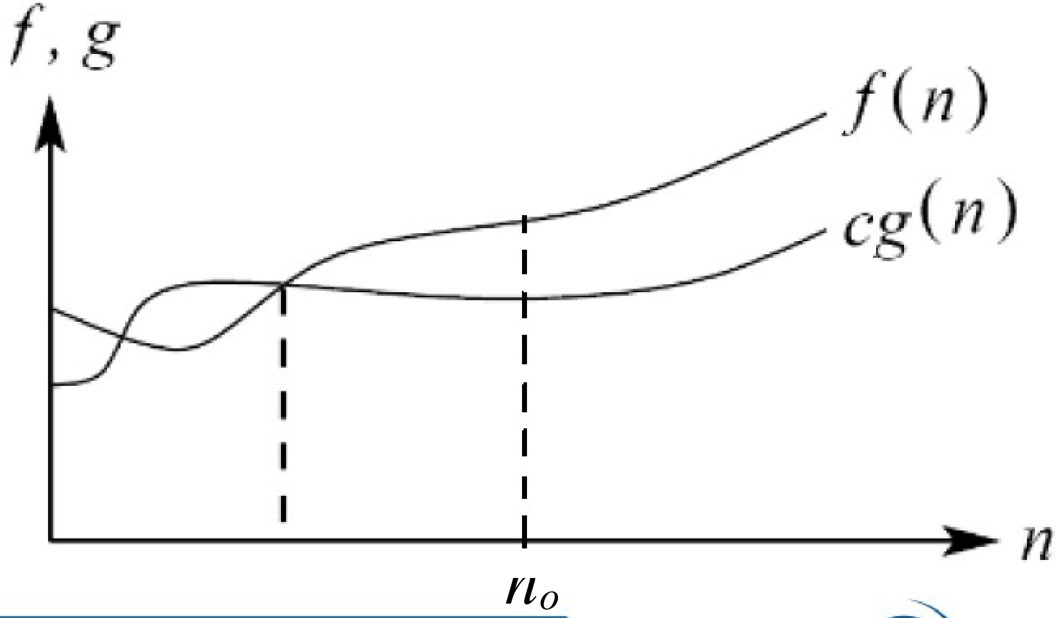
$$\overset{0}{\cancel{\frac{5}{n}}} + \overset{0}{\cancel{\frac{3}{n^2}}} \geq c$$

Logo,
f(n) não é Ω(n^2).
f(n) não é Ω(n^3)...

Notação Ω

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Ω(g(n)) se existem <u>constantes positivas c e n₀</u>, tais que f(n) ≥ c.g(n) ∀ n ≥ n₀.**
- Exemplos: **$f(n) = 5n + 3$**

  $g1(n) = n$ $g2(n) = n^2$ $g3(n) = sqrt(n)$



Demonstração de que f(n) ≥ g3(n):

$5n + 3 \geq 5n \geq 5.sqrt(n)$
$5n + 3n \geq c.sqrt(n)$

Logo, f(n) é Ω(sqrt(n))
f(n) é Ω(log(n)), f(n) é Ω(1)
f(n) não é $\Omega(n^{10})$,
f(n) não é Ω(n log n), f(n) não é $\Omega(n^{1.0001})$, f(n) é $\Omega(n^{0.9999})$.

Notação Ω

- O objetivo de tais demonstrações é justificar por que algo é O(g(n)), por que algo não é O(g(n)); por que algo é Ω(g(n)), por que algo não é Ω(g(n)).
- Futuramente, usaremos tais notações intuitivamente.

n_o

Notação Θ

- A notação **Θ** limita funções inferiormente e superiormente, c1.g(n) ≤ f(n) ≤ c2.g(n).
- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que

  $f(n) = \Theta(g(n))$ [ou $f(n)$ é $\Theta(g(n))$]

  se existem constantes positivas **c_1, c_2 e n_o**, tais que

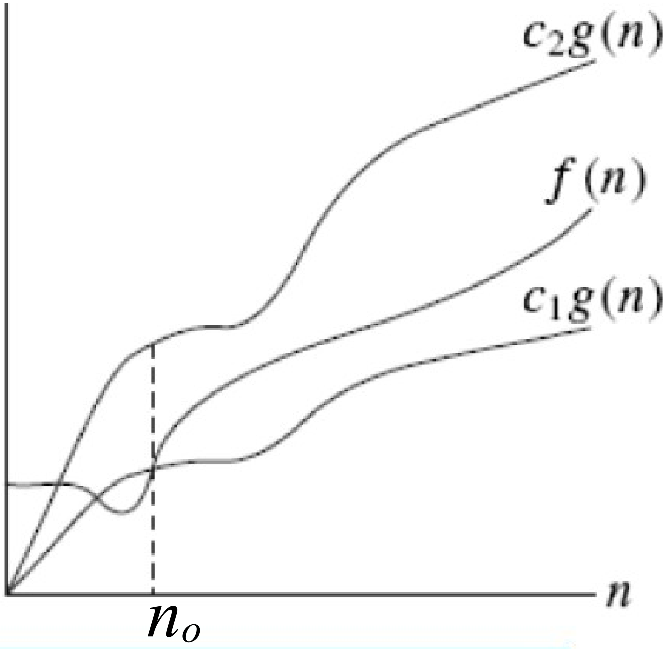
  $$c_1.g(n) \leq f(n) \leq c_2.g(n) \quad \forall \; n \geq n_o$$

- **Obs.:** f(n) é Θ(g(n)) se somente se f(n) é O(g(n)) e é Ω(g(n)).

Notação Θ

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Θ(g(n)) se existem constantes positivas c₁, c₂ e nₒ, tais que c_1.g(n) ≤ f(n) ≤ c_2.g(n) ∀ n ≥ nₒ.**
- Exemplos: **f(n) = 5n + 3**

g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)

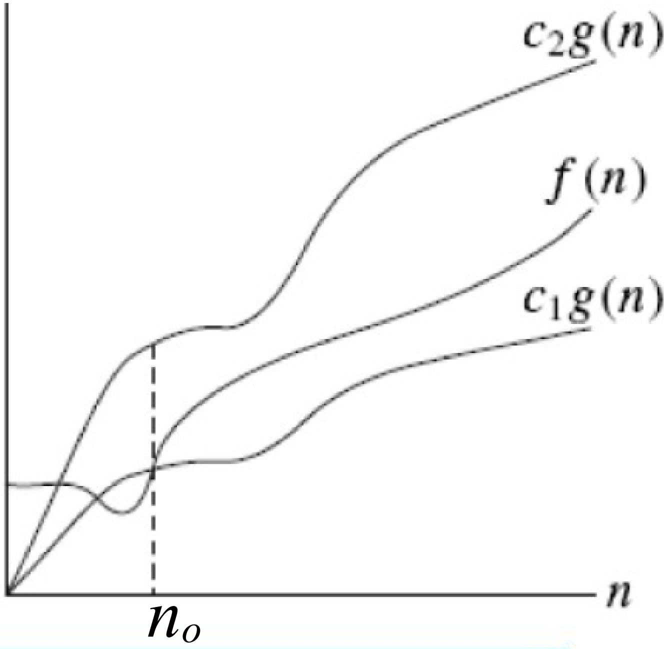


Será possível demonstrar que
c1.n ≤ 5n + 3 ≤ c2.n?

Notação Θ

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Θ(g(n)) se existem constantes positivas c₁, c₂ e nₒ, tais que** **$c_1.g(n) \le f(n) \le c_2.g(n) \quad \forall \quad n \ge n_o$.**
- Exemplos: **$f(n) = 5n + 3$**

  $g1(n) = n$ $\qquad$ $g2(n) = n^2$ $\qquad$ $g3(n) = sqrt(n)$



Demonstração de que

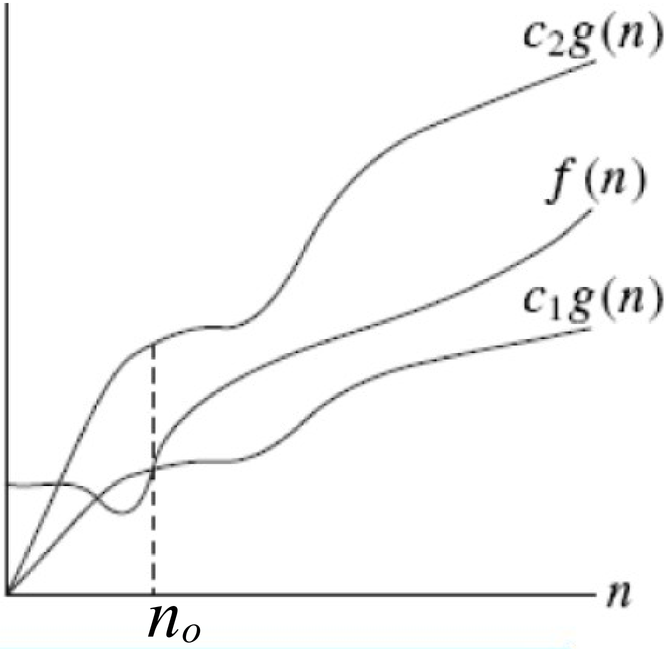
$c_1.g1(n) \le f(n) \le c_2.g1(n)$:

$5n \le 5n + 3 \le 8n$

Notação Θ

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Θ(g(n)) se existem <u>constantes positivas</u> <u>c_1, c_2 e n_o</u>, tais que $c_1.g(n) \leq f(n) \leq c_2.g(n) \quad \forall \ n \geq n_o$.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que
$c_1.g1(n) \leq f(n) \leq c_2.g1(n)$ **:**

$5n \leq 5n + 3 \leq 8n$

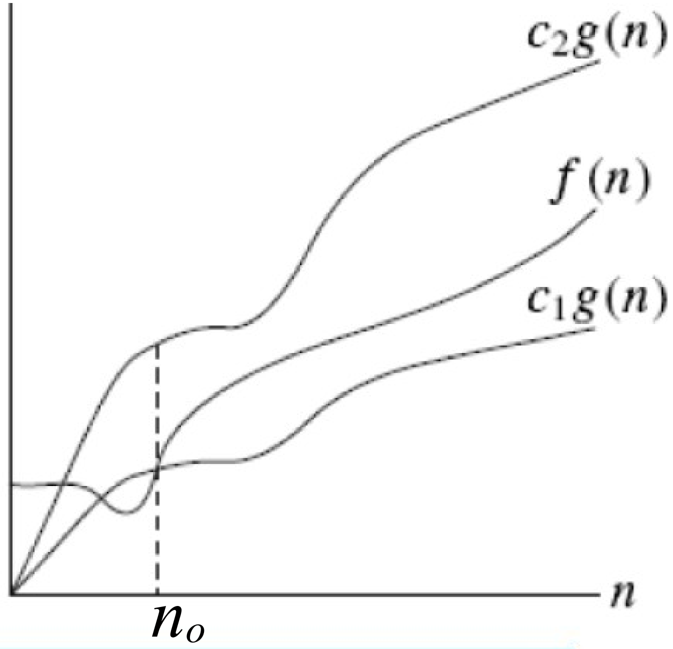
Logo, f(n) é Θ(n).

Demonstração desnecessária, pois já havia sido mostrado que f(n) = 5n + 3 é O(n) e Ω(n).

Notação Θ

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Θ(g(n)) se existem <u>constantes positivas c_1, c_2 e n_o</u>, tais que $c_1.g(n) \leq f(n) \leq c_2.g(n) \quad \forall \; n \geq n_o$.**
- Exemplos: **f(n) = 5n + 3**

g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)

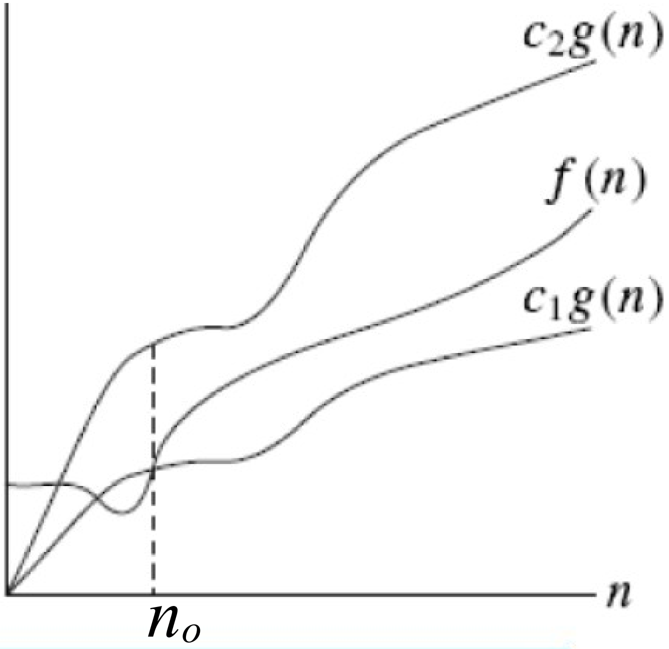


Será possível demonstrar que $c1.n^2 \leq 5n + 3 \leq c2.n^2$?

Notação Θ

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Θ(g(n)) se existem constantes positivas c_1, c_2 e n_o, tais que $c_1.g(n) \leq f(n) \leq c_2.g(n) \quad \forall \; n \geq n_o$.**
- Exemplos: **f(n) = 5n + 3**

  g1(n) = n $g2(n) = n^2$ g3(n) = sqrt(n)



Demonstração de que não é verdade que
$c_1.n^2 \leq f(n) \leq c_2.n^2$ **:**

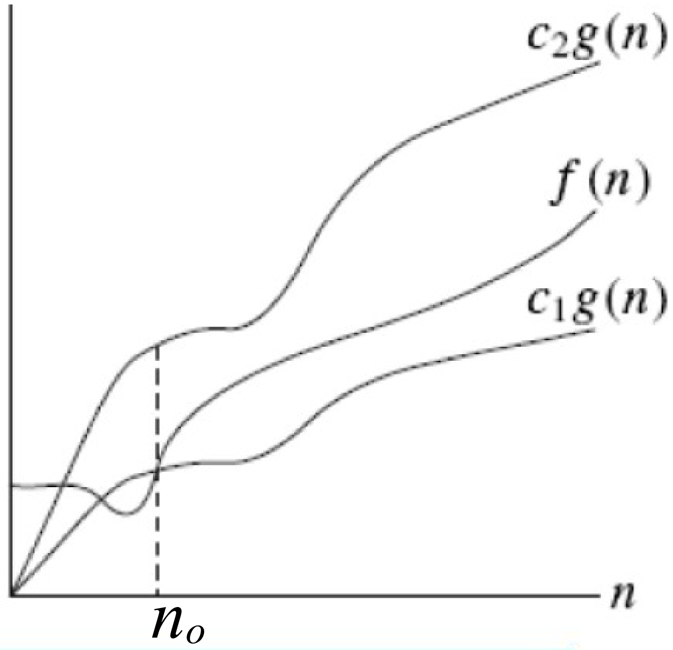
$c1.n^2 \leq 5n + 3 \leq 8n^2$

Logo, f(n) não é $\Theta(n^2)$.

Notação Θ

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Θ(g(n)) se existem __constantes positivas c_1, c_2 e n_o__, tais que $c_1.g(n) \leq f(n) \leq c_2.g(n) \quad \forall \; n \geq n_o$.**
- Exemplos: **f(n) = 5n + 3**

g1(n) = n   g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que não é verdade que

$\cancel{c_1.n^2} \leq f(n) \leq c_2.n^2$:

$c1.n^2 \leq 5n + 3 \leq 8n^2$

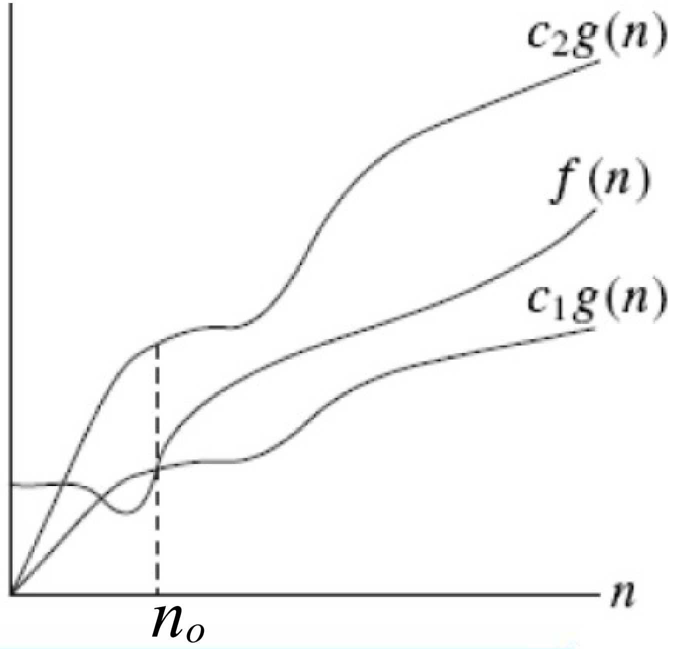
f(n) é O(n^2) somente.

Logo, f(n) não é Θ(n^2).

Notação Θ

- Seja n um inteiro positivo e sejam f(n) e g(n) funções positivas. Dizemos que **f(n) = Θ(g(n)) se existem constantes positivas c_1, c_2 e n_o, tais que $c_1.g(n) \leq f(n) \leq c_2.g(n) \quad \forall \ n \geq n_o$.**
- Exemplos: **f(n) = 5n + 3**

g1(n) = n g2(n) = n^2 g3(n) = sqrt(n)



Demonstração de que não é verdade que
c_1.sqrt(n) ≤ f(n) ≤ c_2.sqrt(n):

Como demonstrando anteriormente, f(n) não é O(sqrt(n)); portanto, f(n) não pode ser Θ(sqrt(n)).

Logo, f(n) não é Θ(sqrt(n)).

Referências Bibliográficas

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; (2002). Algoritmos –Teoria e Prática. Tradução da 2ª edição americana. Rio de Janeiro. Editora Campus.
- TAMASSIA, ROBERTO; GOODRICH, MICHAEL T. (2004). Projeto de Algoritmos -Fundamentos, Análise e Exemplos da Internet.
- ZIVIANI, N. (2007). Projeto e Algoritmos com implementações em Java e C++. São Paulo. Editora Thomson.
- Symbolab (utilizado para gerar alguns gráficos)
  - https://pt.symbolab.com/graphing-calculator

Referências de Material

- Adaptado do material de
  - Professora Carla Negri Lintzmayer da UFABC.