Projeto e Análise de Algoritmos
Notações Assintóticas

Nelson Cruz Sampaio Neto
nelsonneto@ufpa.br

Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Programa de Pós-Graduação em Ciência da Computação

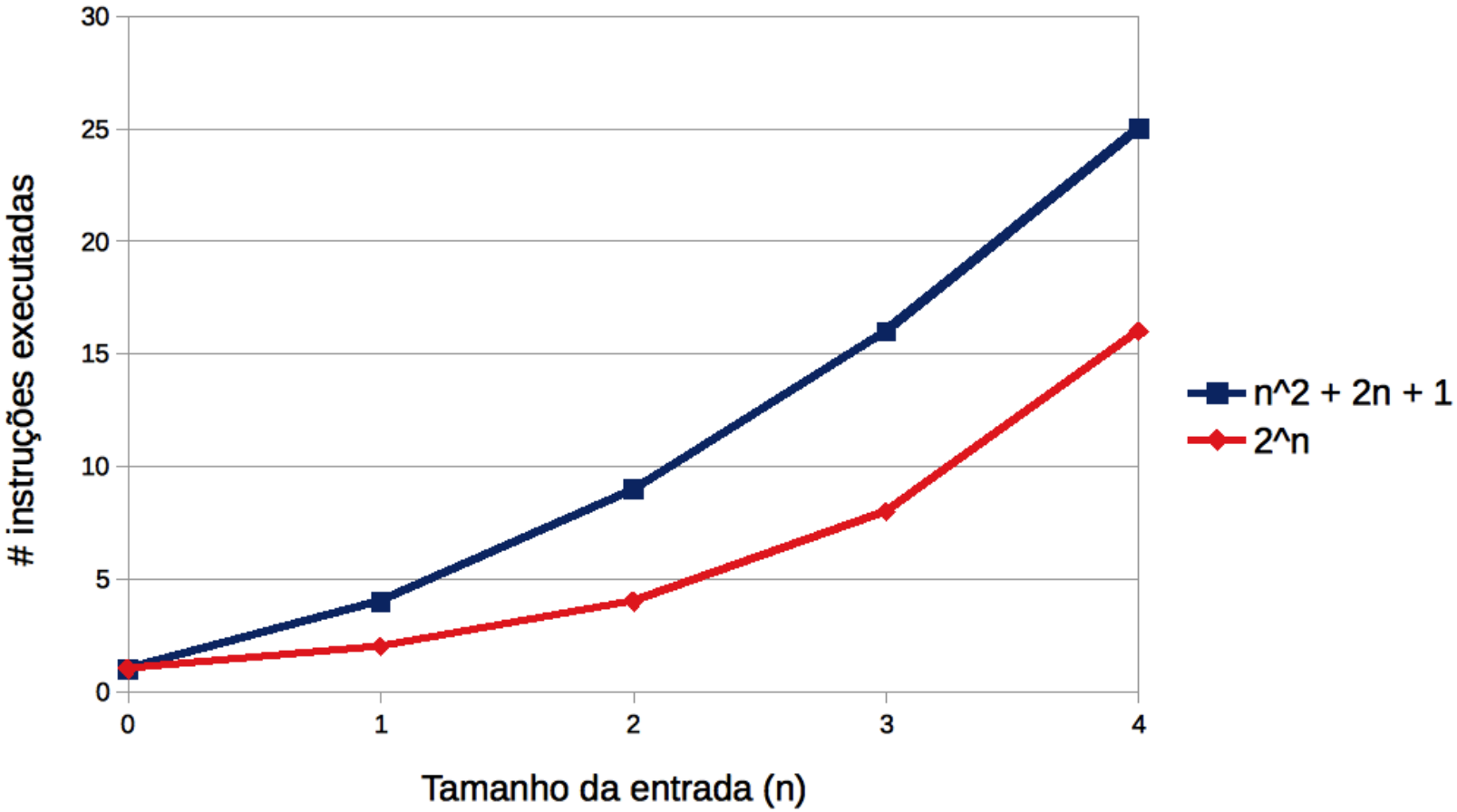
11 de março de 2023

Como comparar algoritmos?

- A unidade de comparação entre dois algoritmos é uma função que calcula o número de instruções no tamanho da entrada.
- Essa função define a **eficiência** ou **complexidade no tempo** do algoritmo.
- Seja a complexidade do algoritmo A: $f(n) = n^2 + 2n + 1$.
- Seja a complexidade do algoritmo B: $g(n) = 2^n$.
- Qual dos dois algoritmos é mais eficiente?

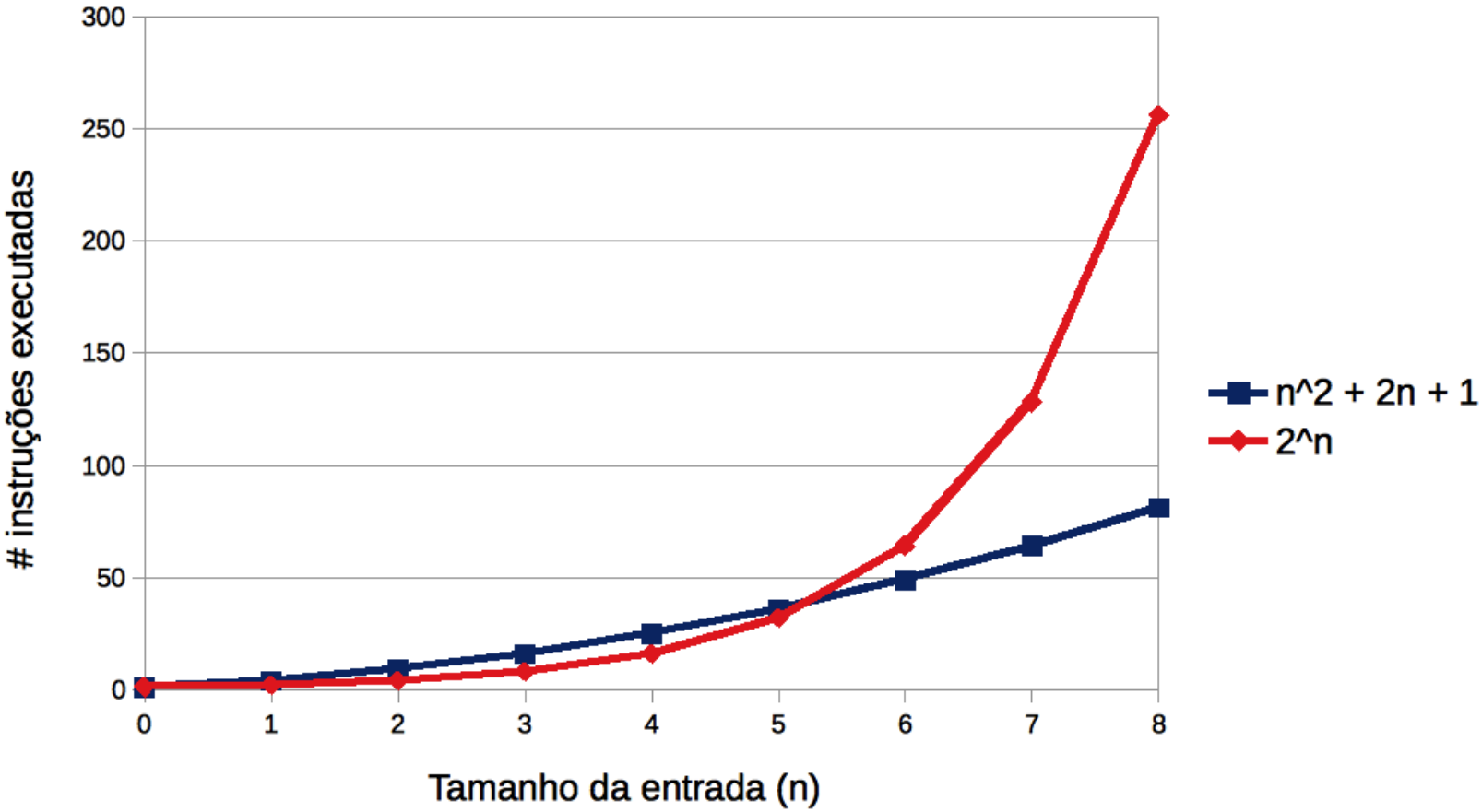
Exemplo

- Primeiramente, analisaremos $f(n)$ e $g(n)$ para $0 \leq n \leq 4$.



Exemplo

- Agora, analisaremos $f(n)$ e $g(n)$ para $0 \leq n \leq 8$.

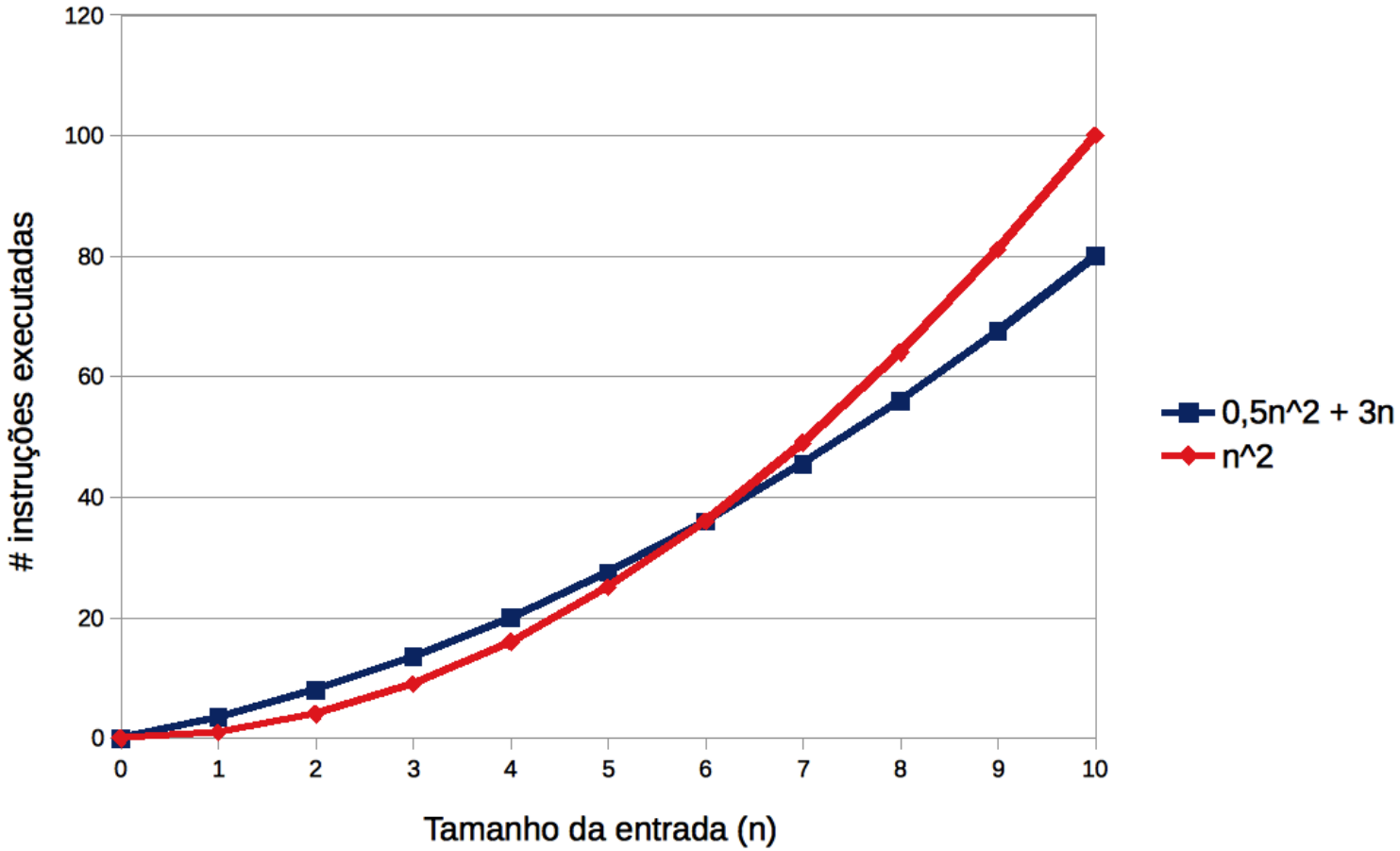


Exemplo

- Conclui-se que $f(n)$ é mais eficiente que $g(n)$ a longo prazo.
- A longo prazo significa "para todo $n \geq 6$", por exemplo.
- É feita uma **análise assintótica** do problema, em outras palavras, considera-se grandes valores nas instâncias.
- O algoritmo B com $g(n) = 2^n$ (exponencial) sequer pode ser considerado um algoritmo eficiente, já que sua complexidade no tempo não é limitada por um polinômio.

Algoritmos equivalentes

- Seja a complexidade do algoritmo A: $f(n) = 0,5n^2 + 3n$.
- Seja a complexidade do algoritmo B: $g(n) = n^2$.



Notação Big-O

- A notação Big-O define o **limite assintótico superior** sobre uma dada função $f(n)$.

$$0 \leq f(n) \leq c\ g(n), \text{ sempre que } n \geq k.$$

$$\text{Ou seja, } f(n) \text{ é } O(g(n)).$$

- c e k são constantes positivas.

- Normalmente usada para limitar o tempo de execução do **pior caso** de um algoritmo.

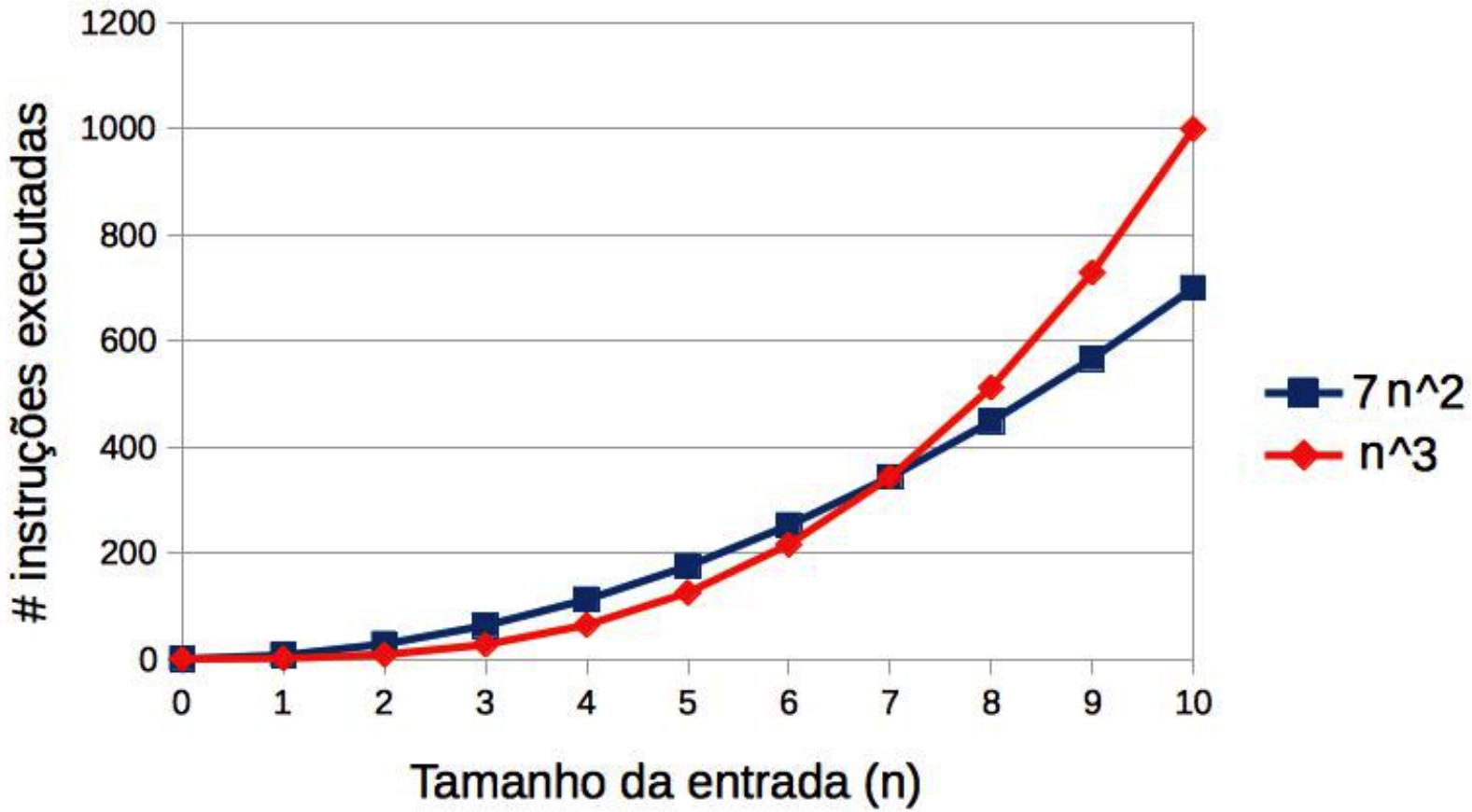
- Por exemplo, $O(n^2)$ é o limite no tempo de execução do pior caso da ordenação por inserção.

Notação Big-O

- Exemplo: Sejam $f(n) = 7n^2$ e $g(n) = n^3$.
- Mostre que $f(n)$ é $O(g(n))$.
- É fácil provar que $0 \leq 7n^2 \leq n^3$, para $n \geq 7$ ($c = 1$ e $k = 7$).
- ($c = 2$ e $k = 4$) também seria uma solução?
- Mostre que $g(n)$ não é $O(f(n))$.
- É fácil provar que n^3 jamais será menor ou igual a $7n^2$ para elevados valores de n.

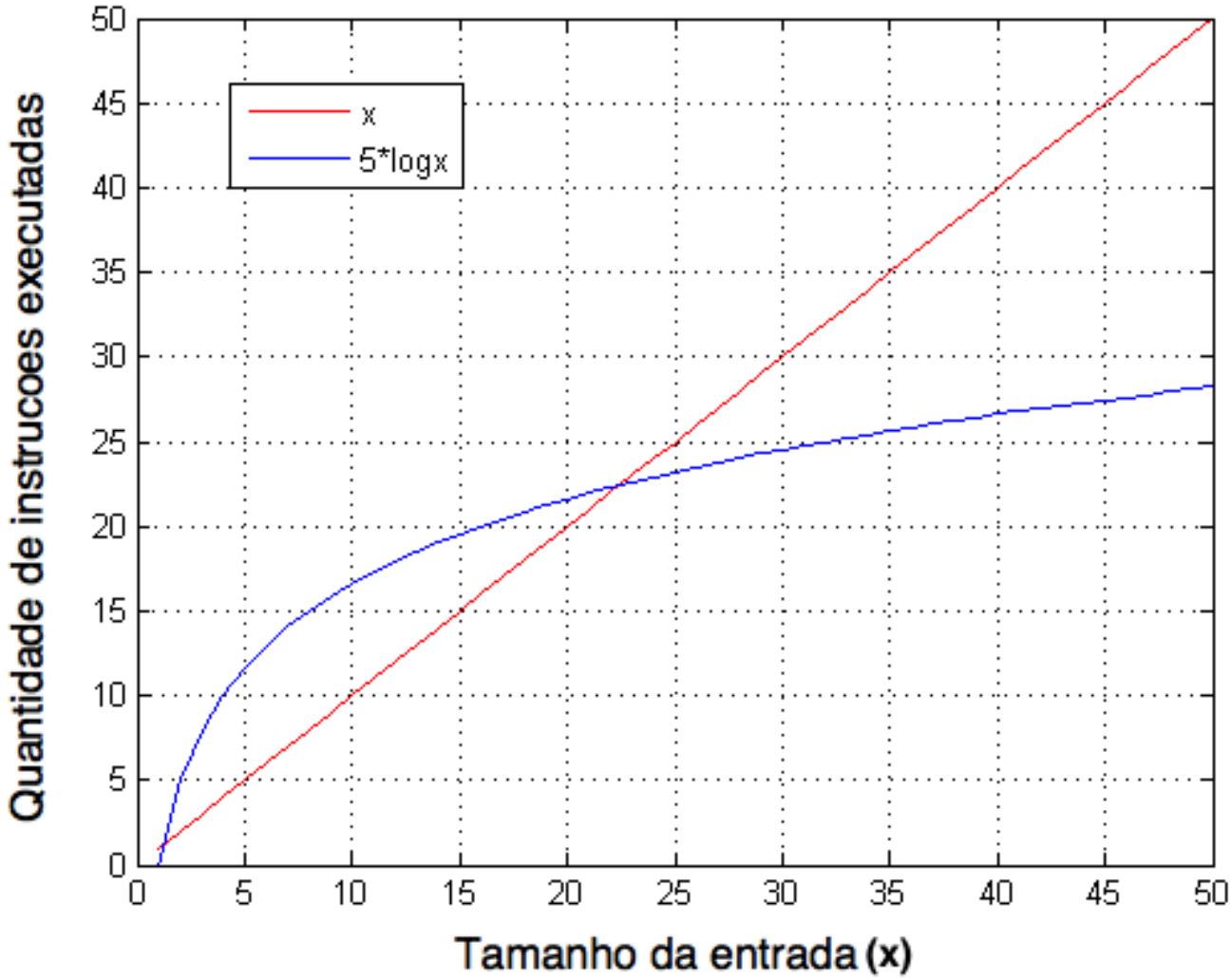
Notação Big-O

- Podemos dizer que $7n^2$ é $O(n^3)$, mas não o contrário.



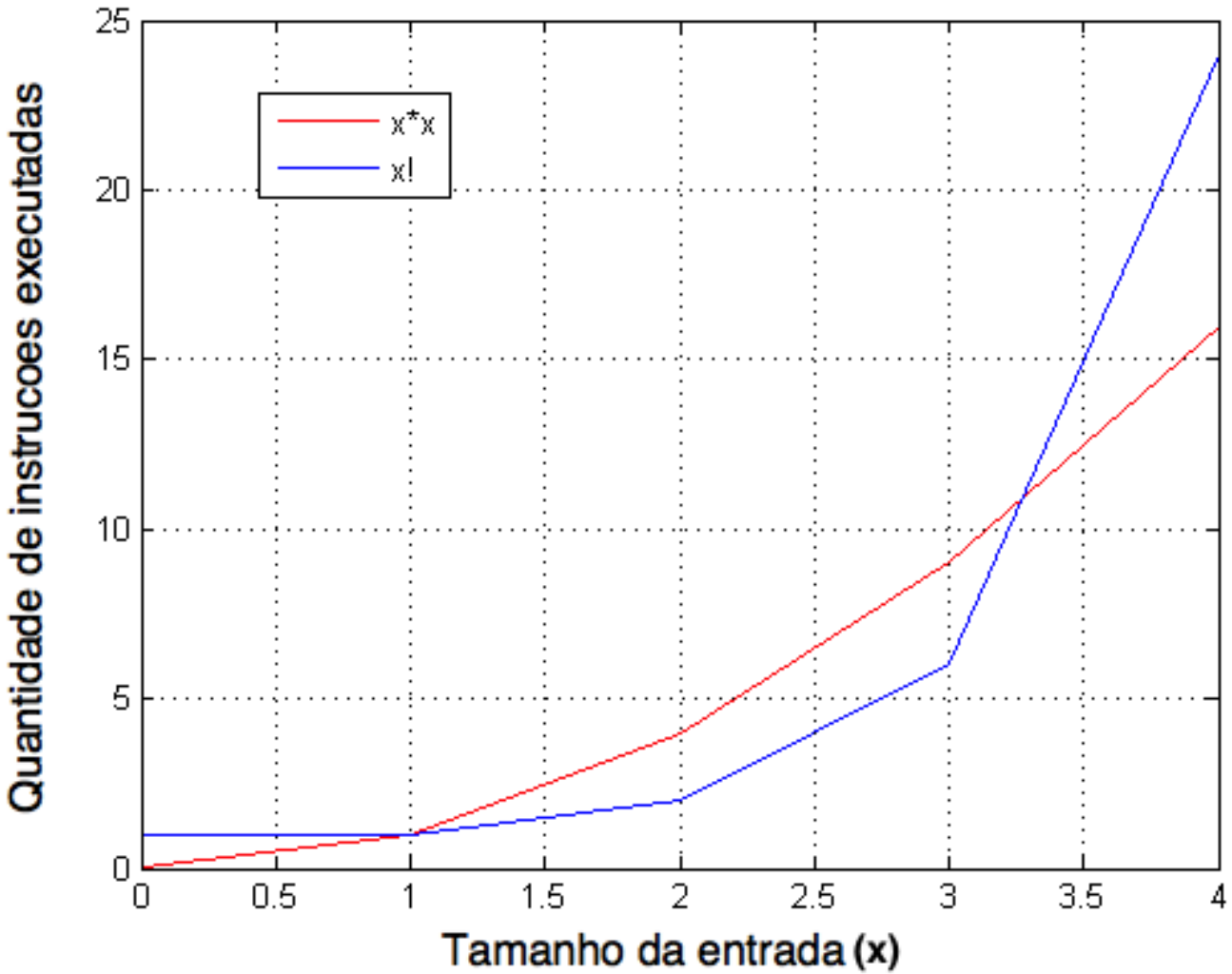
Notação Big-O

- Podemos dizer que $5log_2(x)$ é $O(x)$, mas não o contrário.



Notação Big-O

- Podemos dizer que x^2 é $O(x!)$, mas não o contrário.



Notação Ω

- A notação Ω define o **limite assintótico inferior** sobre uma dada função $f(n)$.

$$0 \leq c\ g(n) \leq f(n), \text{ sempre que } n \geq k.$$

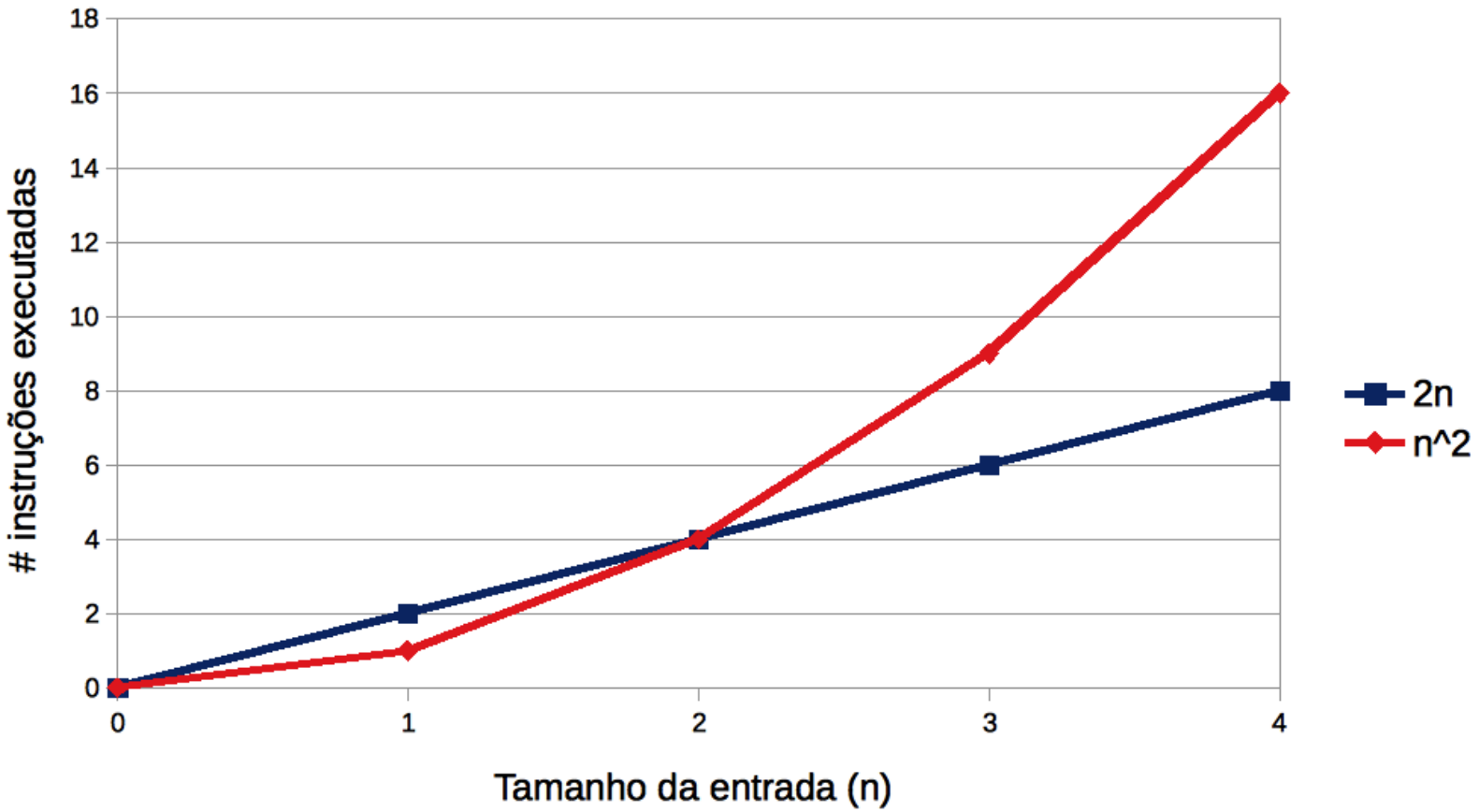
$$\text{Ou seja, } f(n) \text{ é } \Omega(g(n)).$$

- Normalmente usada para limitar o tempo de execução do **melhor caso** de um algoritmo.

- Por exemplo, $\Omega(n)$ é o limite no tempo de execução do melhor caso da ordenação por inserção.

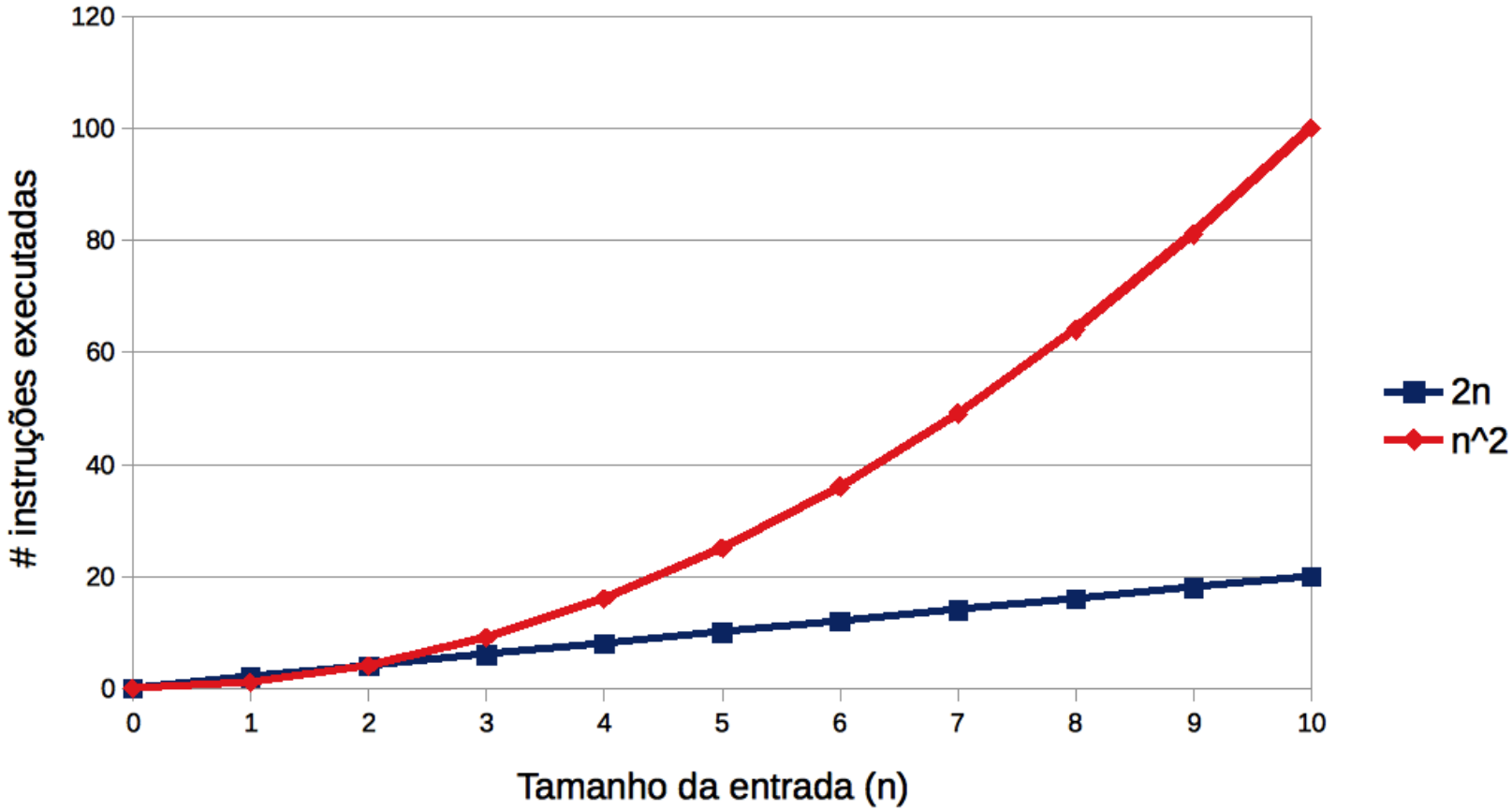
Notação Ω

- Podemos dizer que n^2 é $\Omega(2n)$?



Notação Ω

- Podemos dizer que n^2 é $\Omega(2n)$, mas não o contrário.



Notação Θ

- A notação Θ define o **limite assintótico restrito** sobre uma dada função $f(n)$.
- **Teorema**: Para duas funções $f(n)$ e $g(n)$, dizemos que $f(n)$ é $\Theta(g(n))$ para um valor de n suficientemente grande se e somente se

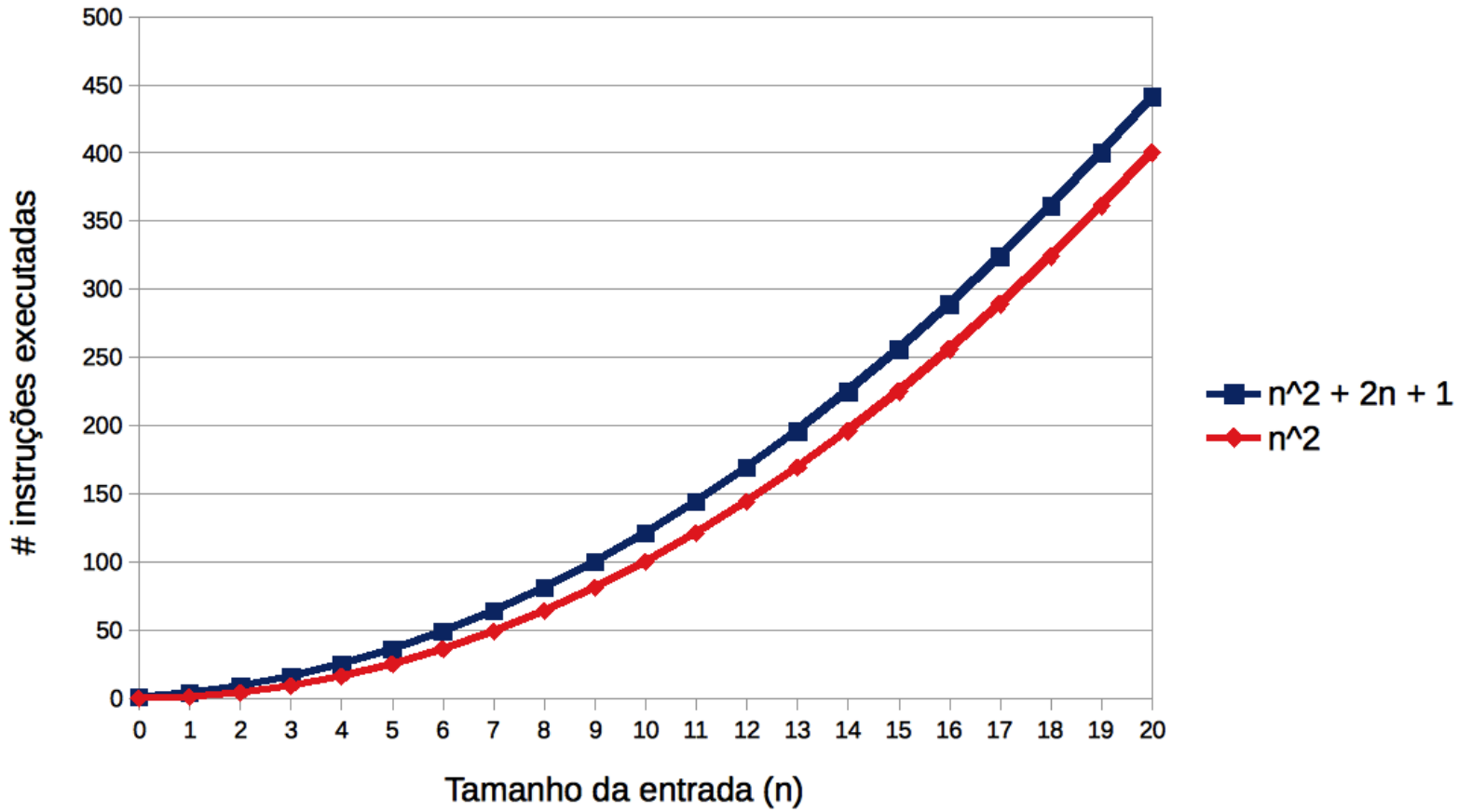
$$f(n) \text{ é } O(g(n)) \text{ e } f(n) \text{ é } \Omega(g(n)).$$

- Podemos dizer que $0,5n^2 + 3n$ é $\Theta(n^2)$.
- Também é fácil verificar que $6n^3$ não é $\Theta(n^2)$.

Notação Θ

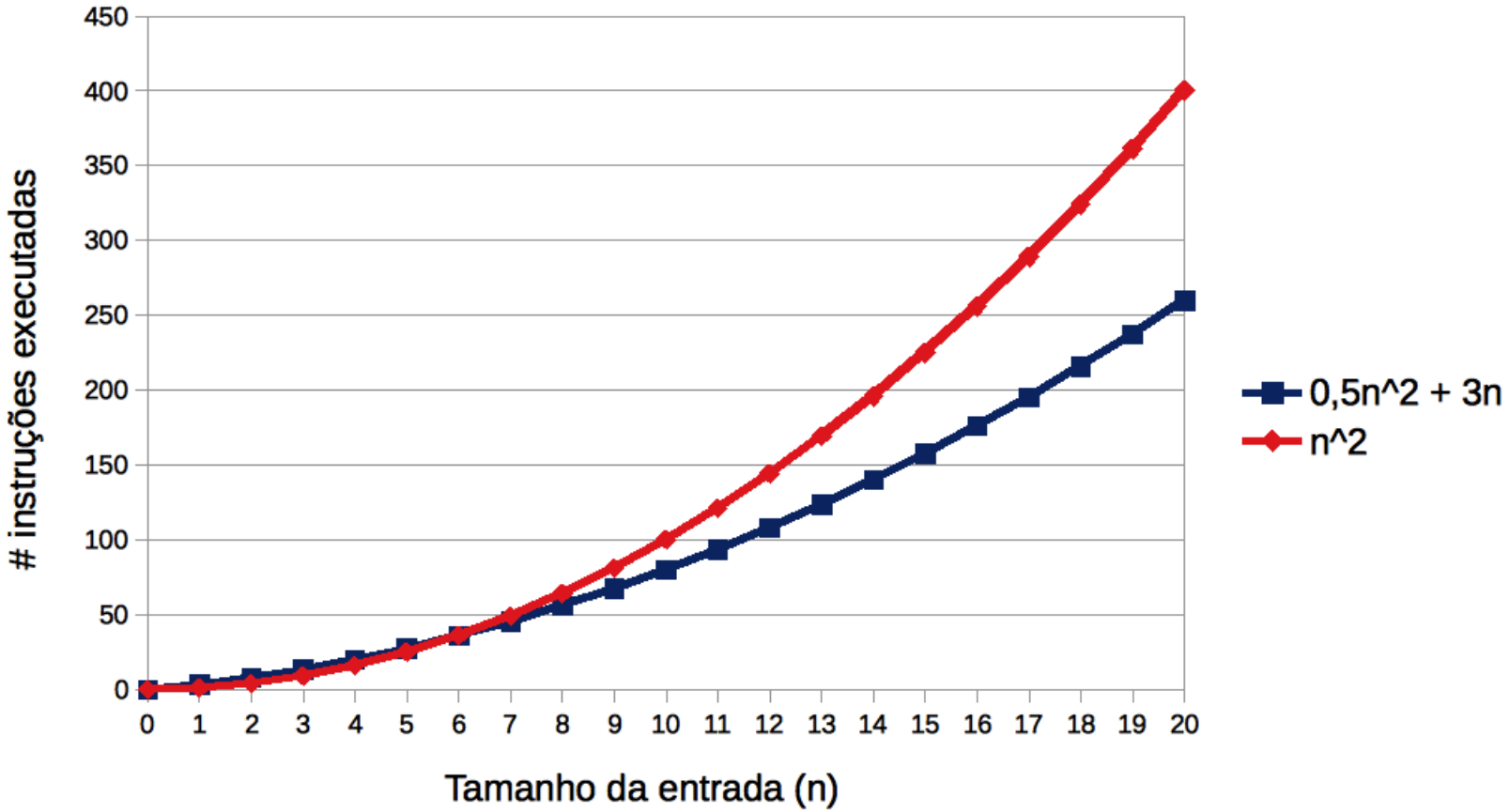
- Exemplo: Sejam $f(n) = n^2 + 2n + 1$ e $g(n) = n^2$.
- É fácil provar que $0 \leq n^2 + 2n + 1 \leq 4n^2$, para $n \geq 1$.
- $f(n)$ parece pior que $g(n)$, mas a longo prazo $f(n)$ é menor ou igual $4g(n)$. Então, $f(n)$ é $O(g(n))$ com $c = 4$ e $k = 1$.
- Também é fácil provar que $0 \leq n^2 \leq n^2 + 2n + 1$, para $n \geq 1$. Então, $g(n)$ é $O(f(n))$ com $c = 1$ e $k = 1$.
- Como "uma é big-O da outra", podemos dizer que elas são Θ, ou ainda, **equivalentes**.

Notação Θ


500
450
400
350
300
250
200
150
100
50
0
instruções executadas
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Tamanho da entrada (n)
n^2 + 2n + 1
n^2

Notação Θ

- Por exemplo, podemos dizer que $0,5n^2 + 3n$ é $\Theta(n^2)$.



Notação Θ

- Mostre que $log_a(n)$ é $\Theta(log_b(n))$.

  i) $0 \leq log_a(n) \leq c\ log_b(n)$

  ii) $0 \leq c'\ log_b(n) \leq log_a(n)$

  i) $log_a(n) \leq c\ log_b(n) = c\ \frac{log_a(n)}{log_a(b)} = \frac{c}{log_a(b)}\ log_a(n)$

Tomando $c = log_a(b)$ e $n \geq 1$, temos o resultado desejado.

ii) Análogo ao anterior.

Notação Little-o

- A notação o define o **limite assintótico superior que não é assintoticamente restrito** sobre uma dada função $f(n)$.

$$0 \leq f(n) < c\ g(n), \text{ sempre que } n \geq k.$$

$$\text{Ou seja, } f(n) \text{ é } o(g(n)).$$

- As definições da notação O e da notação o são semelhantes.
- A principal diferença é que em $f(n) = O(g(n))$, o limite $0 \leq f(n) \leq c\ g(n)$ se mantém válido para **alguma** constante $c > 0$, porém, em $f(n) = o(g(n))$, o limite $0 \leq f(n) < c\ g(n)$ deve se manter válido para **todas** as constantes $c > 0$.
- A notação o não aceita equivalência!
- Por exemplo, $2n$ é $o(n^2)$, mas $2n^2$ não é $o(n^2)$.

Exemplo

- Fazendo uso da definição formal das notações, mostre se a igualdade $10\sqrt{2}^{log(n)} = o(\sqrt{n})$ é verdadeira ou falsa.

$0 \leq 10\sqrt{2}^{log(n)} < c\ \sqrt{n} \quad$ para todo $n \geq k$.

$0 \leq 10n^{log(\sqrt{2})} < c\ \sqrt{n} \quad$ para todo $n \geq k$.

$0 \leq 10n^{0,5} < c\ \sqrt{n} \quad$ para todo $n \geq k$.

$0 \leq 10\sqrt{n} < c\ \sqrt{n} \quad$ para todo $n \geq k$.

$0 \leq 10 < c \quad$ para todo $n \geq k$.

É **falsa** porque não é satisfeita para qualquer $c > 0$.

Notação ω

- A notação ω define o **limite assintótico inferior que não é assintoticamente restrito** sobre uma dada função $f(n)$.

$$0 \leq c\ g(n) < f(n), \text{ sempre que } n \geq k.$$

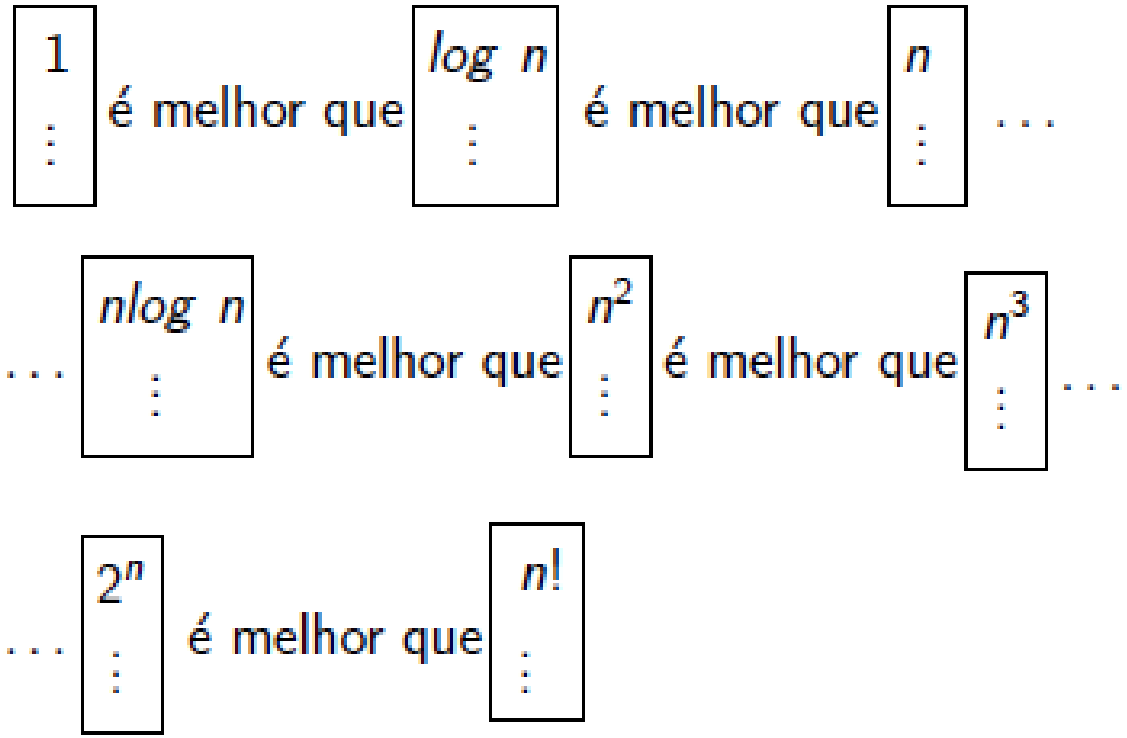
Ou seja, $f(n)$ é $\omega(g(n))$.

- Por analogia, a notação ω está para a notação Ω como a notação o está para a notação O.

- A notação ω não aceita equivalência!

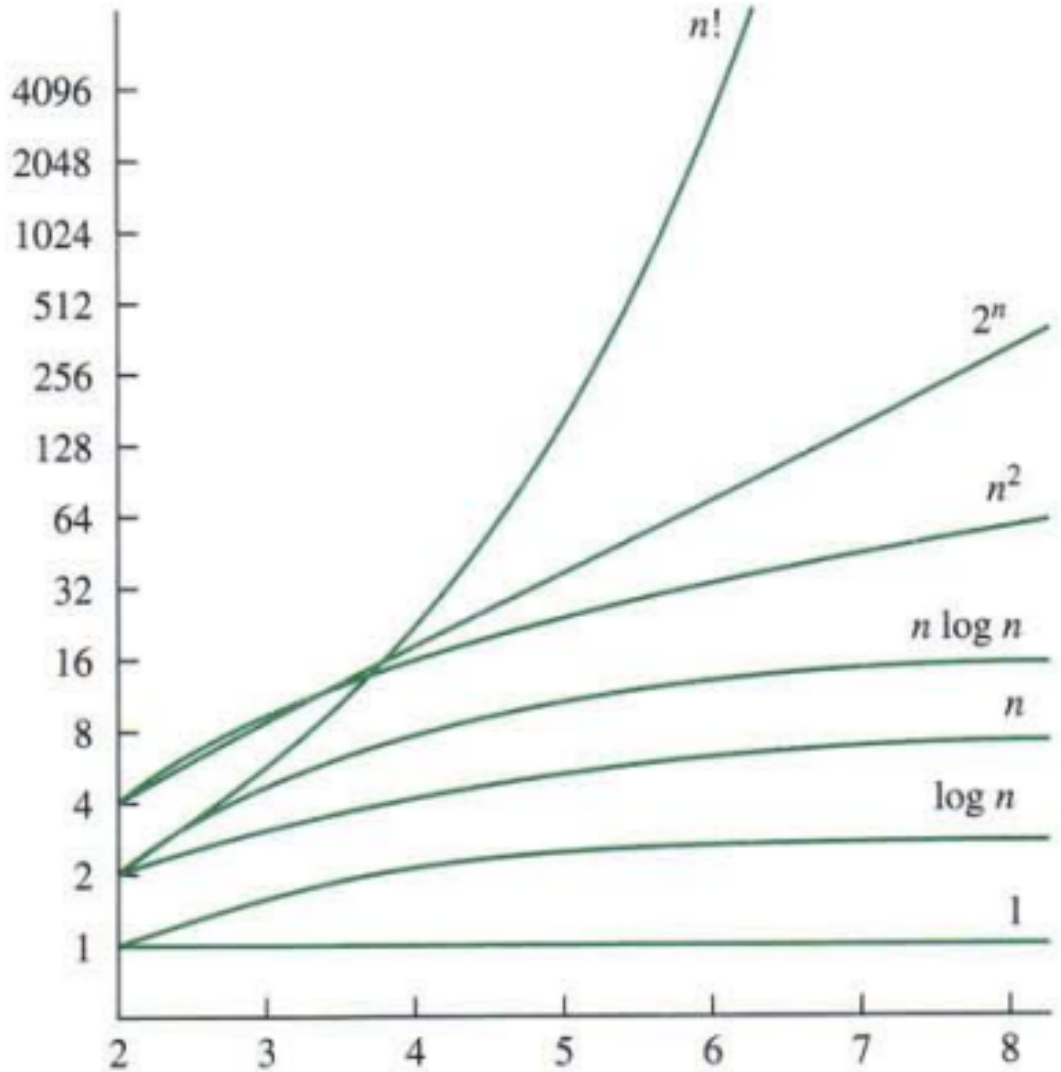
- Por exemplo, $0,5n^2$ é $\omega(n)$, mas $0,5n^2$ não é $\omega(n^2)$.

Classes de complexidade



Obs. Existem mais classes de complexidade que nesta figura! Por exemplo, $\sqrt{n}$, n^4, n^5 etc.

Classes de complexidade



Propriedades

- Na soma de funções, para a notação Big-O, sempre vale a que tem a **maior** complexidade.
  - Qual o Big-O de $f(n) = nlog(n) + 2n + n^2$?
  - Qual o Big-O de $f(n) = n! + 3n^6 + n^3$?
  - Qual o Big-O de $f(n) = (n^4 + 2)(n^5 + 3n^2)$?

Propriedades

- **Transitividade**:

  $f(n) = \Theta(g(n))$ e $g(n) = \Theta(h(n))$ implicam $f(n) = \Theta(h(n))$
  $f(n) = O(g(n))$ e $g(n) = O(h(n))$ implicam $f(n) = O(h(n))$
  $f(n) = \Omega(g(n))$ e $g(n) = \Omega(h(n))$ implicam $f(n) = \Omega(h(n))$
  $f(n) = o(g(n))$ e $g(n) = o(h(n))$ implicam $f(n) = o(h(n))$
  $f(n) = \omega(g(n))$ e $g(n) = \omega(h(n))$ implicam $f(n) = \omega(h(n))$

- **Reflexividade**:

  $f(n) = \Theta(f(n))$
  $f(n) = O(f(n))$
  $f(n) = \Omega(f(n))$

Propriedades

- **Simetria**:

  $f(n) = \Theta(g(n))$ se e somente se $g(n) = \Theta(f(n))$

- **Simetria de transposição**:

  $f(n) = O(g(n))$ se e somente se $g(n) = \Omega(f(n))$
  $f(n) = o(g(n))$ se e somente se $g(n) = \omega(f(n))$

Conclusões

- A notação Big-O é usada para estimar o número de operações que um algoritmo precisa realizar dado o crescimento dos seus dados de entrada.
- Com ela é possível determinar se é prático usar um algoritmo, ou mesmo comparar dois algoritmos para determinar qual é o mais eficiente.
- Por exemplo, se um algoritmo usa $7n^2$ e outro n^3 operações, com a notação Big-O, concluímos que o primeiro realiza um número menor de operações quando n é grande.