Complexidade Assintótica

INF 332 - Projeto e Análise de Algoritmos

José Elias C. Arroyo

Departamento de Informática
Universidade Federal de Viçosa

INF 332 - 2022/2

Outline

Análise Assintótica de Ordens de Grandeza

A análise de eficiência de algoritmos se concentra na ordem de grandeza (**ordem de crescimento**) da função de tempo (que conta o número de execuções da *operação básica*, para entradas suficientemente grande).

Para comparar ordens de crescimento de funções utilizamos as **notações**:

- $O(g(n))$: conjunto de funções que não crescem mais rapidamente que $g(n)$
- $\Theta(g(n))$: conjunto de funções que crescem na mesma ordem que $g(n)$
- $\Omega(g(n))$: conjunto de funções que crescem pelo menos tão rapidamente quanto $g(n)$

Análise Assintótica de Ordens de Grandeza

A análise de eficiência de algoritmos se concentra na ordem de grandeza (**ordem de crescimento**) da função de tempo (que conta o número de execuções da *operação básica*, para entradas suficientemente grande).

Para comparar ordens de crescimento de funções utilizamos as **notações**:

- $O(g(n))$: conjunto de funções que não crescem mais rapidamente que $g(n)$
- $\Theta(g(n))$: conjunto de funções que crescem na mesma ordem que $g(n)$
- $\Omega(g(n))$: conjunto de funções que crescem pelo menos tão rapidamente quanto $g(n)$

Análise Assintótica de Ordens de Grandeza

A análise de eficiência de algoritmos se concentra na ordem de grandeza (**ordem de crescimento**) da função de tempo (que conta o número de execuções da *operação básica*, para entradas suficientemente grande).

Para comparar ordens de crescimento de funções utilizamos as **notações**:

- $O(g(n))$: conjunto de funções que não crescem mais rapidamente que $g(n)$
- $\Theta(g(n))$: conjunto de funções que crescem na mesma ordem que $g(n)$
- $\Omega(g(n))$: conjunto de funções que crescem pelo menos tão rapidamente quanto $g(n)$

Análise Assintótica de Ordens de Grandeza

A análise de eficiência de algoritmos se concentra na ordem de grandeza (**ordem de crescimento**) da função de tempo (que conta o número de execuções da *operação básica*, para entradas suficientemente grande).

Para comparar ordens de crescimento de funções utilizamos as **notações**:

- $O(g(n))$: conjunto de funções que não crescem mais rapidamente que $g(n)$
- $\Theta(g(n))$: conjunto de funções que crescem na mesma ordem que $g(n)$
- $\Omega(g(n))$: conjunto de funções que crescem pelo menos tão rapidamente quanto $g(n)$

Notação O

Informalmente, $O(g(n))$ é a classe ou **conjunto** de todas as funções com uma ordem de crescimento menor ou igual que $g(n)$.

[1] $O(1)$ representa **tempo constante**. Um algoritmo possui tempo constante se ele executa apenas um número constante de "operações básicas". Ou seja, não possui loops dependentes de n (tamanho da entrada).

Notação O

Informalmente, $O(g(n))$ é a classe ou **conjunto** de todas as funções com uma ordem de crescimento menor ou igual que $g(n)$.

As seguintes afirmativas são verdadeiras:

- $n \in O(n^2)$ //n possui ordem de crescimento menor que n^2
- $100n + 5 \in O(n^2)$
- $\frac{1}{2}n(n-1) \in O(n^2)$

[1] $O(1)$ representa **tempo constante**. Um algoritmo possui tempo constante se ele executa apenas um número constante de "operações básicas". Ou seja, não possui loops dependentes de n (tamanho da entrada).

Notação O

Informalmente, $O(g(n))$ é a classe ou **conjunto** de todas as funções com uma ordem de crescimento menor ou igual que $g(n)$.

As seguintes afirmativas são verdadeiras:

- $n \in O(n^2)$ //n possui ordem de crescimento menor que n^2
- $100n + 5 \in O(n^2)$
- $\frac{1}{2}n(n-1) \in O(n^2)$

Por outro lado:

- $n^3 \notin O(n^2)$ //n^3 **não** possui ordem de crescimento **menor** que n^2
- $0.00001n^3 \notin O(n^2)$
- $n^4 + n + 1 \notin O(n^2)$
- $n + 5 \notin O(1)$ [1]

[1] $O(1)$ representa **tempo constante**. Um algoritmo possui tempo constante se ele executa apenas um número constante de "operações básicas". Ou seja, não possui loops dependentes de n (tamanho da entrada).

Notação O

Definição Formal: Notação O

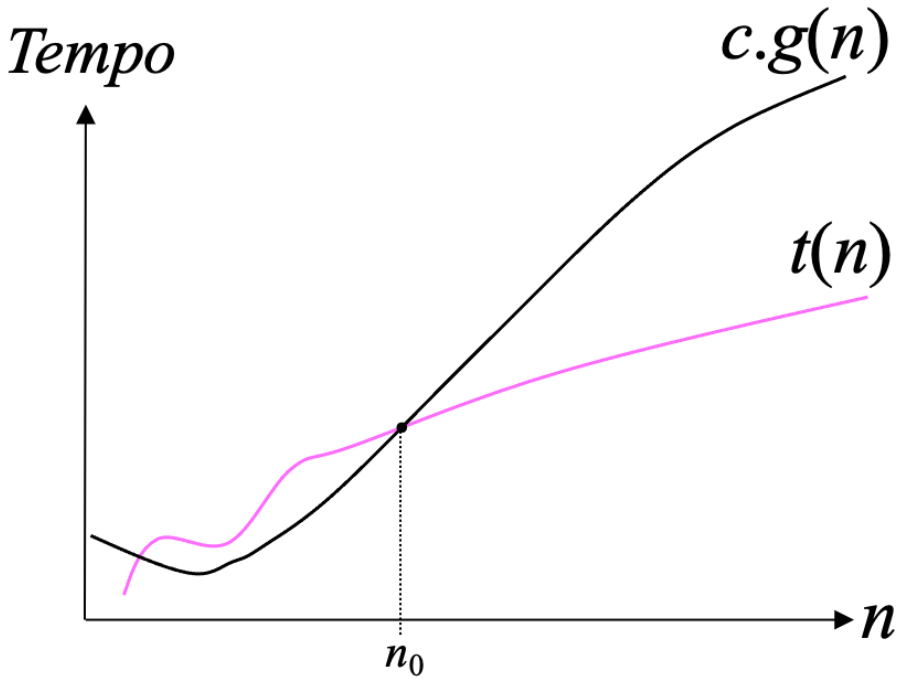
Uma função $t(n)$ está em $O(g(n))$, denotado $t(n) \in O(g(n))$, se $t(n)$ é limitado superiormente por $c \cdot g(n)$ para todo $n \geq n_0$, onde c é uma constante real positiva ($c \in \mathbb{R}^+$) e n_0 um inteiro positivo ($n_0 \in \mathbb{Z}^+$).

Notação O

Definição Formal: Notação O

Uma função $t(n)$ está em $O(g(n))$, denotado $t(n) \in O(g(n))$, se $t(n)$ é limitado superiormente por $c \cdot g(n)$ para todo $n \geq n_0$, onde c é uma constante real positiva ($c \in \mathbb{R}^+$) e n_0 um inteiro positivo ($n_0 \in \mathbb{Z}^+$).

$t(n) \in O(g(n))$ **sss** existem constantes $c > 0$ e $n_0 \geqslant 1$ tq.
$0 \leq t(n) \leq c \cdot g(n)$, $\forall n \geqslant n_0$



Notação O

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Notação O

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n + 5 \leq c.n^2, \forall n \geqslant n_0$.

Notação O

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n + 5 \leq c.n^2, \forall n \geqslant n_0$.

Para $\mathbf{n \geq 5}$: $\quad 100n + \mathbf{5} \leq 100n + \mathbf{n}$

Notação O

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n + 5 \leq c.n^2, \forall n \geqslant n_0$.

Para $\mathbf{n \geq 5}$: $\quad 100n + \mathbf{5} \leq 100n + \mathbf{n} = 101\mathbf{n}$

Notação O

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n + 5 \leq c.n^2, \forall n \geqslant n_0$.

Para $\mathbf{n \geq 5}$: $\quad 100n + \mathbf{5} \leq 100n + \mathbf{n} = 101\mathbf{n} \leq 101\mathbf{n}^2$

Notação O

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n + 5 \leq c.n^2, \forall n \geqslant n_0$.

Para $\mathbf{n \geq 5}$: $\quad 100n + \mathbf{5} \leq 100n + \mathbf{n} = 101\mathbf{n} \leq 101\mathbf{n}^2$

Ou seja, existem $c = 101$ e $n_0 = 5$ tq. $100n + 5 \leq c.n^2, \forall n \geq 5$

Notação O

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n + 5 \leq c.n^2, \forall n \geqslant n_0$.

Para $\mathbf{n \geq 5}$: $\quad 100n + \mathbf{5} \leq 100n + \mathbf{n} = 101\mathbf{n} \leq 101\mathbf{n}^2$

Ou seja, existem $c = 101$ e $n_0 = 5$ tq. $100n + 5 \leq c.n^2, \forall n \geq 5$

Portanto, $100n + 5 \in O(n^2)$

Notação O

A definição da notação O nos dá liberdade para escolher os valores específicos para c e n_0. Ou seja, nas provas podem ser utilizados diferentes valores para c e n_0.

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Outra maneira de provar $100n + 5 \in O(n^2)$:

Notação O

A definição da notação O nos dá liberdade para escolher os valores específicos para c e n_0. Ou seja, nas provas podem ser utilizados diferentes valores para c e n_0.

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Outra maneira de provar $100n + 5 \in O(n^2)$:
$100n + 5 \leq 100n + 5n$ (para todo $n \geq 1$)

Notação O

A definição da notação O nos dá liberdade para escolher os valores específicos para c e n_0. Ou seja, nas provas podem ser utilizados diferentes valores para c e n_0.

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Outra maneira de provar $100n + 5 \in O(n^2)$:
$100n + 5 \leq 100n + 5n$ (para todo $n \geq 1$)
$100n + 5 \leq 105n$

Notação O

A definição da notação O nos dá liberdade para escolher os valores específicos para c e n_0. Ou seja, nas provas podem ser utilizados diferentes valores para c e n_0.

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Outra maneira de provar $100n + 5 \in O(n^2)$:
$100n + 5 \leq 100n + 5n$ (para todo $n \geq 1$)
$100n + 5 \leq 105n \leq 105n^2$

Notação O

A definição da notação O nos dá liberdade para escolher os valores específicos para c e n_0. Ou seja, nas provas podem ser utilizados diferentes valores para c e n_0.

Exemplo 1: prove que $100n + 5 \in O(n^2)$

Outra maneira de provar $100n + 5 \in O(n^2)$:
$100n + 5 \leq 100n + 5n$ (para todo $n \geq 1$)
$100n + 5 \leq 105n \leq 105n^2$

Portanto, $100n + 5 \in O(n^2)$ para $c = 105$ e $n_0 = 1$.

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^2, \forall n \geqslant n_0$.

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^2, \forall n \geqslant n_0$.

Para $n = 1$: $100n^2 + 4n + 2 = 100(1) + 4(1) + 2 = 106$

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^2, \forall n \geqslant n_0$.

Para $n = 1$: $100n^2 + 4n + 2 = 100(1) + 4(1) + 2 = 106$

Então $100n^2 + 4n + 2 \leq 106n^2, \forall n \geqslant 1$

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^2, \forall n \geqslant n_0$.

Para $n = 1$: $100n^2 + 4n + 2 = 100(1) + 4(1) + 2 = 106$

Então $100n^2 + 4n + 2 \leq 106n^2, \forall n \geqslant 1$

Portanto, $100n^2 + 4n + 2 \in O(n^2)$

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Outra maneira de provar:

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Outra maneira de provar:

Para $n \geq 1$: $\quad 100n^2 = 100n^2$

Para $n \geq 5$: $\quad 4n + 2 \leq n^2$

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Outra maneira de provar:

Para $n \geq 1$: $\quad 100n^2 = 100n^2$

Para $n \geq 5$: $\quad 4n + 2 \leq n^2$

Então, $100n^2 + 4n + 2 \leq 101n^2, \forall n \geqslant 5$.

Notação O

Exemplo 2: prove que $100n^2 + 4n + 2 \in O(n^2)$

Outra maneira de provar:

Para $n \geq 1$: $\quad 100n^2 = 100n^2$

Para $n \geq 5$: $\quad 4n + 2 \leq n^2$

Então, $100n^2 + 4n + 2 \leq 101n^2, \forall n \geqslant 5$.

Portanto, $100n^2 + 4n + 2 \in O(n^2)$

Notação O

Exemplo 3: Prove que: $100n^2 + 4n + 2 \in O(n^4)$

Notação O

Exemplo 3: Prove que: $100n^2 + 4n + 2 \in O(n^4)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^4, \forall n \geqslant n_0$.

Notação O

Exemplo 3: Prove que: $100n^2 + 4n + 2 \in O(n^4)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^4, \forall n \geqslant n_0$.

Para $n \geq 1$: $\quad 100n^2 \leq 100n^4$

Notação O

Exemplo 3: Prove que: $100n^2 + 4n + 2 \in O(n^4)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^4, \forall n \geqslant n_0$.

Para $n \geq 1$: $\quad 100n^2 \leq 100n^4$

Para $n \geq 2$: $\quad 4n + 2 \leq n^4$

Notação O

Exemplo 3: Prove que: $100n^2 + 4n + 2 \in O(n^4)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^4, \forall n \geqslant n_0$.

Para $n \geq 1$: $\quad 100n^2 \leq 100n^4$

Para $n \geq 2$: $\quad 4n + 2 \leq n^4$

Então, $100n^2 + 4n + 2 \leq 101n^4, \forall n \geqslant 2$.

Notação O

Exemplo 3: Prove que: $100n^2 + 4n + 2 \in O(n^4)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$100n^2 + 4n + 2 \leq c.n^4, \forall n \geqslant n_0$.

Para $n \geq 1$: $\quad 100n^2 \leq 100n^4$

Para $n \geq 2$: $\quad 4n + 2 \leq n^4$

Então, $100n^2 + 4n + 2 \leq 101n^4, \forall n \geqslant 2$.

Portanto, $100n^2 + 4n + 2 \in O(n^4)$

Notação O

Exercício: Prove que: $1000n^2 + 100n - 6 \in O(n^2)$

Notação O

Exemplo 5: prove que: $3 \times 2^n + n^2 \in O(2^n)$

Notação O

Exemplo 5: prove que: $3 \times 2^n + n^2 \in O(2^n)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$3 \times 2^n + n^2 \leq c.2^n, \forall n \geqslant n_0$.

Notação O

Exemplo 5: prove que: $3 \times 2^n + n^2 \in O(2^n)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$3 \times 2^n + n^2 \leq c.2^n, \forall n \geqslant n_0$.

Para $n \geq 1$: $\quad 3.2^n = 3.2^n$

Notação O

Exemplo 5: prove que: $3 \times 2^n + n^2 \in O(2^n)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$3 \times 2^n + n^2 \leq c.2^n, \forall n \geqslant n_0$.

Para $n \geq 1$: $3.2^n = 3.2^n$

Para $n \geq 4$: $n^2 \leq 2^n$

Notação O

Exemplo 5: prove que: $3 \times 2^n + n^2 \in O(2^n)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$3 \times 2^n + n^2 \leq c.2^n, \forall n \geqslant n_0$.

Para $n \geq 1$: $\quad 3.2^n = 3.2^n$

Para $n \geq 4$: $\quad n^2 \leq 2^n$

Então, $3.2^n + n^2 \leq 4.2^n, \forall n \geqslant 4$.

Notação O

Exemplo 5: prove que: $3 \times 2^n + n^2 \in O(2^n)$

Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
$3 \times 2^n + n^2 \leq c.2^n, \forall n \geqslant n_0$.

Para $n \geq 1$: $\quad 3.2^n = 3.2^n$

Para $n \geq 4$: $\quad n^2 \leq 2^n$

Então, $3.2^n + n^2 \leq 4.2^n, \forall n \geqslant 4$.

Portanto, $3.2^n + n^2 \in O(2^n)$

Notação O

Note que, as seguintes afirmações são corretas:

- $\frac{(n+1)(n-1)}{2} \in O(n^3)$
- $\frac{(n+1)(n-1)}{2} \in O(n^4)$
- $\frac{(n+1)(n-1)}{2} \in O(2^n)$

Em análise de algoritmos, estamos interessados no **menor limite superior** possível de $\frac{(n+1)(n-1)}{2}$.

Ou seja, $\frac{(n+1)(n-1)}{2} \in O(n^2)$

Não é usual escrever $\frac{(n+1)(n-1)}{2} \in O(n^3)$

Notação O

Note que, as seguintes afirmações são corretas:

- $\frac{(n+1)(n-1)}{2} \in O(n^3)$
- $\frac{(n+1)(n-1)}{2} \in O(n^4)$
- $\frac{(n+1)(n-1)}{2} \in O(2^n)$

Em análise de algoritmos, estamos interessados no **menor limite superior** possível de $\frac{(n+1)(n-1)}{2}$.

Ou seja, $\frac{(n+1)(n-1)}{2} \in O(n^2)$

Não é usual escrever $\frac{(n+1)(n-1)}{2} \in O(n^3)$

Notação O

Note que, as seguintes afirmações são corretas:

- $\frac{(n+1)(n-1)}{2} \in O(n^3)$
- $\frac{(n+1)(n-1)}{2} \in O(n^4)$
- $\frac{(n+1)(n-1)}{2} \in O(2^n)$

Em análise de algoritmos, estamos interessados no **menor limite superior** possível de $\frac{(n+1)(n-1)}{2}$.

Ou seja, $\frac{(n+1)(n-1)}{2} \in O(n^2)$

Não é usual escrever $\frac{(n+1)(n-1)}{2} \in O(n^3)$

Notação Ω

Informalmente, $\Omega(g(n))$ é o **conjunto** de todas as funções com ordem de grandeza maior ou igual que $g(n)$.

Por exemplo:

- $\frac{1}{2}n(n-1) \in \Omega(n^2)$
- $100n + 5 \notin \Omega(n^2)$
- $n^3 \in \Omega(n^3)$
- $n^3 \in \Omega(n^2)$
- $n^3 \in \Omega(n)$

n, n^2, n^3 são **limites inferiores** de n^3.

No entanto, estamos interessados no **maior limite inferior** de n^3.

Ou seja, $n^3 \in \Omega(n^3)$

Notação Ω

Informalmente, $\Omega(g(n))$ é o **conjunto** de todas as funções com ordem de grandeza maior ou igual que $g(n)$.

Por exemplo:

- $\frac{1}{2}n(n-1) \in \Omega(n^2)$
- $100n + 5 \notin \Omega(n^2)$
- $n^3 \in \Omega(n^3)$
- $n^3 \in \Omega(n^2)$
- $n^3 \in \Omega(n)$

n, n^2, n^3 são **limites inferiores** de n^3.

No entanto, estamos interessados no **maior limite inferior** de n^3.

Ou seja, $n^3 \in \Omega(n^3)$

Notação Ω

Informalmente, $\Omega(g(n))$ é o **conjunto** de todas as funções com ordem de grandeza maior ou igual que $g(n)$.

Por exemplo:

- $\frac{1}{2}n(n-1) \in \Omega(n^2)$
- $100n + 5 \notin \Omega(n^2)$
- $n^3 \in \Omega(n^3)$
- $n^3 \in \Omega(n^2)$
- $n^3 \in \Omega(n)$

n, n^2, n^3 são **limites inferiores** de n^3.

No entanto, estamos interessados no **maior limite inferior** de n^3.

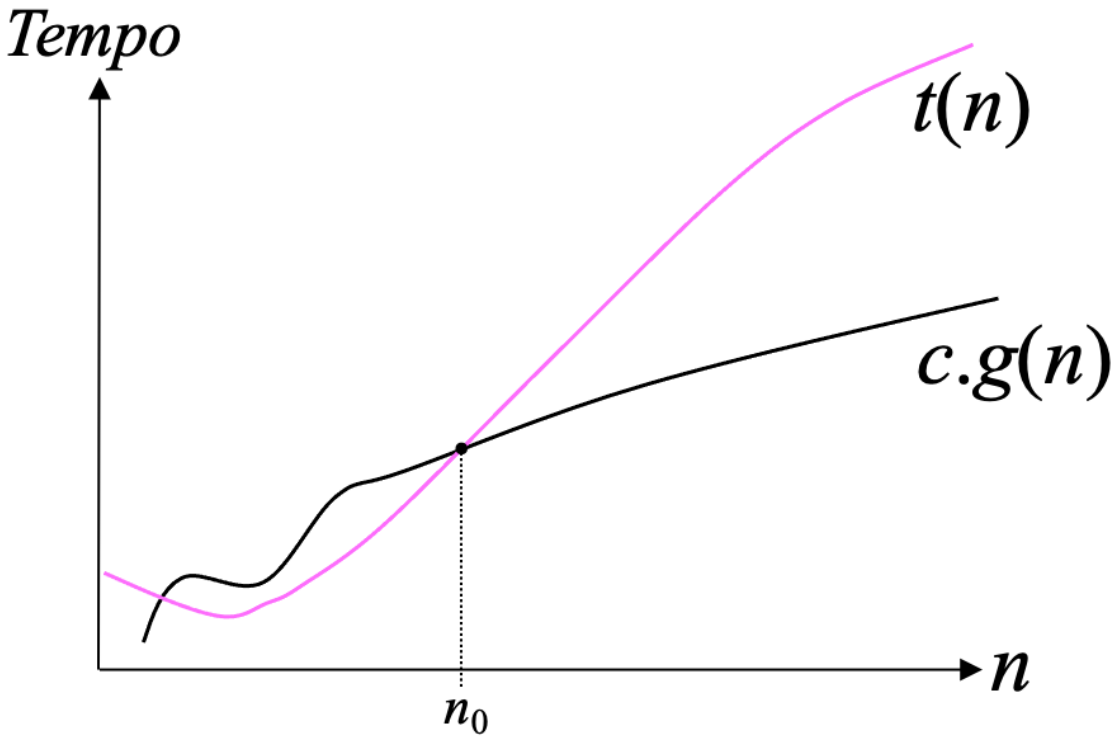
Ou seja, $n^3 \in \Omega(n^3)$

Notação Ω

Definição Formal: Notação Ω

$t(n) \in \Omega(g(n))$, se $t(n)$ é limitado inferiormente por $c \cdot g(n)$ para todo $n \geq n_0$, onde $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$.

$t(n) \in \Omega(g(n))$ **sss** existem constantes $c > 0$ e $n_0 \geqslant 1$, tq.
$t(n) \geqslant c \cdot g(n) \geqslant 0$, $\forall n \geqslant n_0$



Notação Ω

Exemplo 1: prove que $n^3 - 50 \in \Omega(n^2)$

Notação Ω

Exemplo 1: prove que $n^3 - 50 \in \Omega(n^2)$

- Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
  $n^3 - 50 \geq c.n^2, \quad \forall n \geqslant n_0$

Notação Ω

Exemplo 1: prove que $n^3 - 50 \in \Omega(n^2)$

- Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
  $n^3 - 50 \geq c.n^2, \quad \forall n \geqslant n_0$
- Note que para $n = 5$: $\quad n^3 - 50 = 75 \quad$ e $\quad n^2 = 25$

Notação Ω

Exemplo 1: prove que $n^3 - 50 \in \Omega(n^2)$

- Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
  $n^3 - 50 \geq c.n^2, \quad \forall n \geqslant n_0$
- Note que para $n = 5$: $\quad n^3 - 50 = 75 \quad$ e $\quad n^2 = 25$
- Ou seja, $n^3 - 50 \geq 1.n^2, \quad \forall n \geqslant 5$

Notação Ω

Exemplo 1: prove que $n^3 - 50 \in \Omega(n^2)$

- Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq. $n^3 - 50 \geq c.n^2, \quad \forall n \geqslant n_0$
- Note que para $n = 5$: $\quad n^3 - 50 = 75 \quad$ e $\quad n^2 = 25$
- Ou seja, $n^3 - 50 \geq 1.n^2, \quad \forall n \geqslant 5$
- Logo, para $c = 1$ e $n_0 = 5$: $n^3 - 50 \in \Omega(n^2)$

Notação Ω

Exemplo 2: prove que $n^2 - 2n \in \Omega(n^2)$

Notação Ω

Exemplo 2: prove que $n^2 - 2n \in \Omega(n^2)$

- Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
  $n^2 - 2n \geq c.n^2, \quad \forall n \geqslant n_0$

Notação Ω

Exemplo 2: prove que $n^2 - 2n \in \Omega(n^2)$

- Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
  $n^2 - 2n \geq c.n^2, \quad \forall n \geqslant n_0$

- Note que para $n = 3$: $\quad n^2 - 2n = \mathbf{3} \quad$ e $\quad n^2 = \mathbf{9}$

Notação Ω

Exemplo 2: prove que $n^2 - 2n \in \Omega(n^2)$

- Temos que provar que existem $c \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$ tq.
  $n^2 - 2n \geq c.n^2, \quad \forall n \geqslant n_0$
- Note que para $n = 3$: $\quad n^2 - 2n = \mathbf{3} \quad$ e $\quad n^2 = \mathbf{9}$
- Escolher: $c = 3/9 = 1/3$
- Então: $n^2 - 2n \geq \frac{1}{3}.n^2, \quad \forall n \geqslant 3$

Notação Ω

Exercícios: Prove que:

- $2^n - n^2 \in \Omega(n^2)$
- $n^4 - 8n \in \Omega(n^4)$.
- $3n + 4 \in \Omega(1)$

Notação Θ

Informalmente, $\Theta(g(n))$ é o **conjunto** de todas as funções com a mesma ordem de crescimento de $g(n)$.

Por exemplo:

- $n^2 + n \in \Theta(n^2)$
- $\frac{1}{2}n(n-1) \in \Theta(n^2)$
- $7n^2 + 5 \notin \Theta(n)$

Notação Θ

Informalmente, $\Theta(g(n))$ é o **conjunto** de todas as funções com a mesma ordem de crescimento de $g(n)$.

Por exemplo:

- $n^2 + n \in \Theta(n^2)$
- $\frac{1}{2}n(n-1) \in \Theta(n^2)$
- $7n^2 + 5 \notin \Theta(n)$

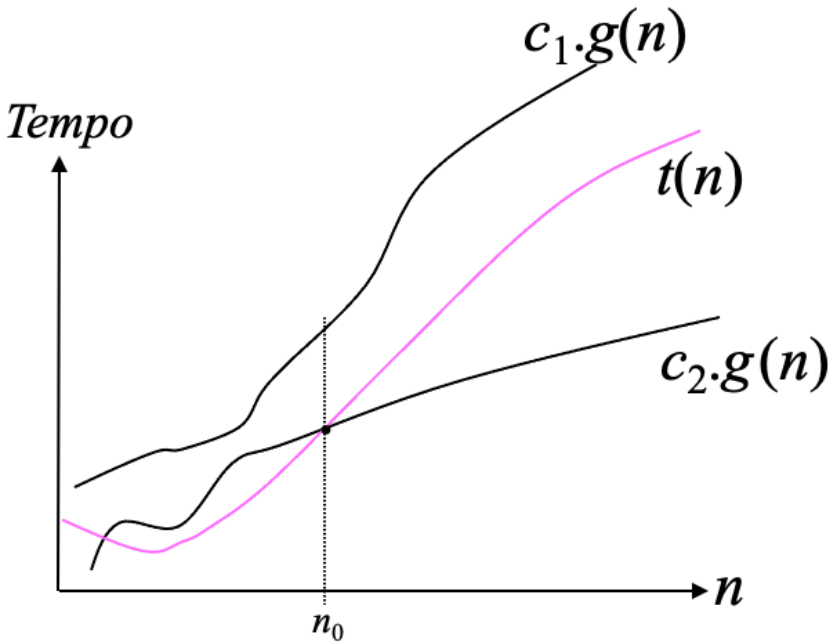
$t(n) \in \Theta(g(n))$ sss $t(n) \in O(g(n))$ e $t(n) \in \Omega(g(n))$.

Notação Θ

Definição Formal: Notação Θ

$t(n) \in \Theta(g(n))$, se $t(n)$ é limitado superiormente por $c_1 \cdot g(n)$ e inferiormente por $c_2 \cdot g(n)$ para todo $n \geq n_0$, onde c_1 e $c_2 \in \mathbb{R}^+$ e $n_0 \in \mathbb{Z}^+$.

$t(n) \in \Theta(g(n))$ **sss** existem constantes reais $c_1, c_2 > 0$ e $n_0 \geqslant 1$ tq. $0 \leq c_2 \cdot g(n) \leq t(n) \leq c_1 \cdot g(n)$, $\forall n \geqslant n_0$



Notação Θ

Exemplo 1: prove que $\frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$

Notação Θ

Exemplo 1: prove que $\frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$

Primeiro provar que existem c_1 e n_0 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \leq c_1 n^2, \forall n \geq n_0$ (limite superior):

Notação Θ

Exemplo 1: prove que $\frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$

Primeiro provar que existem c_1 e n_0 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \leq c_1 n^2, \forall n \geq n_0$ (limite superior):

- $\frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2$ (para todo $n \geq 1$)

Notação Θ

Exemplo 1: prove que $\frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$

Primeiro provar que existem c_1 e n_0 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \leq c_1 n^2, \forall n \geq n_0$ (limite superior):

- $\frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2$ (para todo $n \geq 1$)

Em seguida provar que existe c_2 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \geq c_2 n^2, \forall n \geq n_0$ (limite inferior):

Notação Θ

Exemplo 1: prove que $\frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$

Primeiro provar que existem c_1 e n_0 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \leq c_1 n^2, \forall n \geq n_0$ (limite superior):

- $\frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2$ (para todo $n \geq 1$)

Em seguida provar que existe c_2 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \geq c_2 n^2, \forall n \geq n_0$ (limite inferior):

- Note que: $-\frac{1}{2}n \geq -\frac{1}{2}n\frac{1}{2}n = -\frac{1}{4}n^2$, para todo $n \geq 2$,

Notação Θ

Exemplo 1: prove que $\frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$

Primeiro provar que existem c_1 e n_0 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \leq c_1 n^2, \forall n \geq n_0$ (limite superior):

- $\frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2$ (para todo $n \geq 1$)

Em seguida provar que existe c_2 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \geq c_2 n^2, \forall n \geq n_0$ (limite inferior):

- Note que: $-\frac{1}{2}n \geq -\frac{1}{2}n\frac{1}{2}n = -\frac{1}{4}n^2$, para todo $n \geq 2$,
- Então: $\frac{1}{2}n^2 - \frac{1}{2}n \geq \frac{1}{2}n^2 - \frac{1}{4}n^2 = \frac{1}{4}n^2$, para todo $n \geq 2$

Notação Θ

Exemplo 1: prove que $\frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$

Primeiro provar que existem c_1 e n_0 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \leq c_1 n^2, \forall n \geq n_0$ (limite superior):

- $\frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2$ (para todo $n \geq 1$)

Em seguida provar que existe c_2 tq. $\frac{1}{2}n^2 - \frac{1}{2}n \geq c_2 n^2, \forall n \geq n_0$ (limite inferior):

- Note que: $-\frac{1}{2}n \geq -\frac{1}{2}n\frac{1}{2}n = -\frac{1}{4}n^2$, para todo $n \geq 2$,
- Então: $\frac{1}{2}n^2 - \frac{1}{2}n \geq \frac{1}{2}n^2 - \frac{1}{4}n^2 = \frac{1}{4}n^2$, para todo $n \geq 2$

Portanto, existem $c_1 = \frac{1}{2}$, $c_2 = \frac{1}{4}$ e $n_0 = 2$, tq.

$c_2 n^2 \leq \frac{1}{2}n^2 - \frac{1}{2}n \leq c_1 n^2, \forall n \geq n_0$.

Usando limites para comparar ordens de grandeza

O uso de **limites** pode ser mais conveniente para **comparar a ordem de grandeza** de duas funções específicas:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0, & f(n) \text{ possui ordem de grandeza menor que } g(n) \\ c > 0, & f(n) \text{ possui ordem de grandeza igual a } g(n) \\ \infty, & f(n) \text{ possui ordem de grandeza maior que } g(n) \end{cases}$$

Usando limites para comparar ordens de grandeza

- Se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, então $f(n) \in O(g(n))$
- Se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0$, então $f(n) \in \Theta(g(n))$
- Se $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, então $f(n) \in \Omega(g(n))$

Usando limites para comparar ordens de grandeza

Exemplo #1: Compare as grandezas de $100n + 5$ e n^2

Usando limites para comparar ordens de grandeza

Exemplo #1: Compare as grandezas de $100n + 5$ e n^2

Mostrar que $100n + 5$ possui ordem de grandeza **menor** que n^2,

ou seja, $\lim_{n \to \infty} \frac{100n + 5}{n^2} = 0$

Usando limites para comparar ordens de grandeza

Exemplo #1: Compare as grandezas de $100n + 5$ e n^2

Mostrar que $100n + 5$ possui ordem de grandeza **menor** que n^2,

ou seja, $\lim_{n \to \infty} \frac{100n + 5}{n^2} = 0$

$$\lim_{n \to \infty} \frac{100n + 5}{n^2} = \lim_{n \to \infty} (\frac{100n}{n^2} + \frac{5}{n^2}) = \lim_{n \to \infty} (\frac{100}{n} + \frac{5}{n^2}) =$$

Usando limites para comparar ordens de grandeza

Exemplo #1: Compare as grandezas de $100n + 5$ e n^2

Mostrar que $100n + 5$ possui ordem de grandeza **menor** que n^2,

ou seja, $\lim_{n\to\infty} \frac{100n+5}{n^2} = 0$

$$\lim_{n\to\infty} \frac{100n+5}{n^2} = \lim_{n\to\infty} (\frac{100n}{n^2} + \frac{5}{n^2}) = \lim_{n\to\infty} (\frac{100}{n} + \frac{5}{n^2}) =$$

$$= 100 \lim_{n\to\infty} \frac{1}{n} + 5 \lim_{n\to\infty} \frac{1}{n^2} = 0$$

Usando limites para comparar ordens de grandeza

Exemplo #1: Compare as grandezas de $100n + 5$ e n^2

Mostrar que $100n + 5$ possui ordem de grandeza **menor** que n^2,

ou seja, $\lim_{n\to\infty} \frac{100n+5}{n^2} = 0$

$$\lim_{n\to\infty} \frac{100n+5}{n^2} = \lim_{n\to\infty} (\frac{100n}{n^2} + \frac{5}{n^2}) = \lim_{n\to\infty} (\frac{100}{n} + \frac{5}{n^2}) =$$

$$= 100 \lim_{n\to\infty} \frac{1}{n} + 5 \lim_{n\to\infty} \frac{1}{n^2} = 0$$

- Então: $100n + 5 \in O(n^2)$

Usando limites para comparar ordens de grandeza

Exemplo #2: Compare as grandezas de $\frac{1}{2}n(n-1)$ e n^2

Usando limites para comparar ordens de grandeza

Exemplo #2: Compare as grandezas de $\frac{1}{2}n(n-1)$ e n^2

Mostrar que $\frac{1}{2}n(n-1)$ e n^2 possuem a mesma ordem de grandeza, ou seja, $\lim_{n\to\infty} \frac{\frac{1}{2}n(n-1)}{n^2} = c > 0$

Usando limites para comparar ordens de grandeza

Exemplo #2: Compare as grandezas de $\frac{1}{2}n(n-1)$ e n^2

Mostrar que $\frac{1}{2}n(n-1)$ e n^2 possuem a mesma ordem de grandeza, ou seja, $\lim_{n\to\infty} \frac{\frac{1}{2}n(n-1)}{n^2} = c > 0$

$$\lim_{n\to\infty} \frac{\frac{1}{2}n(n-1)}{n^2} = \frac{1}{2}\lim_{n\to\infty} \frac{n^2-n}{n^2} = \frac{1}{2}\lim_{n\to\infty}(1-\frac{1}{n}) = \frac{1}{2}$$

Usando limites para comparar ordens de grandeza

Exemplo #2: Compare as grandezas de $\frac{1}{2}n(n-1)$ e n^2

Mostrar que $\frac{1}{2}n(n-1)$ e n^2 possuem a mesma ordem de grandeza, ou seja, $\lim_{n\to\infty} \frac{\frac{1}{2}n(n-1)}{n^2} = c > 0$

$$\lim_{n\to\infty} \frac{\frac{1}{2}n(n-1)}{n^2} = \frac{1}{2}\lim_{n\to\infty} \frac{n^2-n}{n^2} = \frac{1}{2}\lim_{n\to\infty}(1-\frac{1}{n}) = \frac{1}{2}$$

- As duas funções possuem a mesma ordem de grandeza
- Então: $\frac{1}{2}n(n-1) \in \Theta(n^2)$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2 + 5n + 2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2 + 5n + 2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

$n \in \Omega(\log_2 n)$: Mostrar que $\lim_{n\to\infty} \frac{n}{\log_2 n} = \infty$

(RL'H): $\lim_{n\to\infty} \frac{n}{\log_2 n} = \lim_{n\to\infty} \frac{1}{1/(n \ln 2)} = \lim_{n\to\infty} n \ln 2 = \ln 2 \lim_{n\to\infty} n = \infty$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2 + 5n + 2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

$n \in \Omega(\log_2 n)$: Mostrar que $\lim_{n\to\infty} \frac{n}{\log_2 n} = \infty$

(RL'H): $\lim_{n\to\infty} \frac{n}{\log_2 n} = \lim_{n\to\infty} \frac{1}{1/(n \ln 2)} = \lim_{n\to\infty} n \ln 2 = \ln 2 \lim_{n\to\infty} n = \infty$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2 + 5n + 2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

$\sqrt{n} \in \Omega(\log_2 n)$: Mostrar que $\lim_{n\to\infty} \frac{\sqrt{n}}{\log_2 n} = \infty$

(RL'H): $\lim_{n\to\infty} \frac{\sqrt{n}}{\log_2 n} = \lim_{n\to\infty} \frac{1/(2\sqrt{n})}{1/(n \ln 2)} = \lim_{n\to\infty} \frac{\ln 2.n}{2\sqrt{n}} = \frac{\ln 2}{2} \lim_{n\to\infty} \frac{n}{\sqrt{n}}$

(RL'H): $\lim_{n\to\infty} \frac{n}{\sqrt{n}} = \lim_{n\to\infty} \frac{1}{1/(2\sqrt{n})} = \lim_{n\to\infty} 2\sqrt{n} = \infty$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2+5n+2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

$\sqrt{n} \in \Omega(\log_2 n)$: Mostrar que $\lim_{n\to\infty} \frac{\sqrt{n}}{\log_2 n} = \infty$

(RL'H): $\lim_{n\to\infty} \frac{\sqrt{n}}{\log_2 n} = \lim_{n\to\infty} \frac{1/(2\sqrt{n})}{1/(n \ln 2)} = \lim_{n\to\infty} \frac{\ln 2.n}{2\sqrt{n}} = \frac{\ln 2}{2} \lim_{n\to\infty} \frac{n}{\sqrt{n}}$

(RL'H): $\lim_{n\to\infty} \frac{n}{\sqrt{n}} = \lim_{n\to\infty} \frac{1}{1/(2\sqrt{n})} = \lim_{n\to\infty} 2\sqrt{n} = \infty$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2+5n+2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

$\sqrt{n} \in \Omega(\log_2 n)$: Mostrar que $\lim_{n\to\infty} \frac{\sqrt{n}}{\log_2 n} = \infty$

(RL'H): $\lim_{n\to\infty} \frac{\sqrt{n}}{\log_2 n} = \lim_{n\to\infty} \frac{1/(2\sqrt{n})}{1/(n \ln 2)} = \lim_{n\to\infty} \frac{\ln 2.n}{2\sqrt{n}} = \frac{\ln 2}{2} \lim_{n\to\infty} \frac{n}{\sqrt{n}}$

(RL'H): $\lim_{n\to\infty} \frac{n}{\sqrt{n}} = \lim_{n\to\infty} \frac{1}{1/(2\sqrt{n})} = \lim_{n\to\infty} 2\sqrt{n} = \infty$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2 + 5n + 2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

$n! \in \Omega(2^n)$: Mostrar que $\lim_{n\to\infty} \frac{n!}{2^n} = \infty$

$n! = \sqrt{2\pi n}.(\frac{n}{e})^n$ Fórmula de Stirling.

$$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi}.\sqrt{n}.(\frac{n}{e})^n}{2^n} = \sqrt{2\pi} \lim_{n\to\infty} \sqrt{n}.(\frac{n}{2e})^n = \sqrt{2\pi}.\infty.\infty = \infty$$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2 + 5n + 2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

$n! \in \Omega(2^n)$: Mostrar que $\lim_{n\to\infty} \frac{n!}{2^n} = \infty$

$n! = \sqrt{2\pi n}.(\frac{n}{e})^n$ Fórmula de Stirling.

$$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi}.\sqrt{n}.(\frac{n}{e})^n}{2^n} = \sqrt{2\pi} \lim_{n\to\infty} \sqrt{n}.(\frac{n}{2e})^n = \sqrt{2\pi}.\infty.\infty = \infty$$

Usando limites para comparar ordens de grandeza

Exercícios: Usando limites mostrar:

- $2^n \in O(3^n)$
- $n \in \Omega(\log_2 n)$
- $\sqrt{n} \in \Omega(\log_2 n)$
- $\sqrt{4n^2 + 5n + 2} \in \Omega(n^2)$
- $n! \in \Omega(2^n)$
- $n! \in O(n^n)$

$n! \in \Omega(2^n)$: Mostrar que $\lim_{n\to\infty} \frac{n!}{2^n} = \infty$

$n! = \sqrt{2\pi n}.(\frac{n}{e})^n$ Fórmula de Stirling.

$$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi}.\sqrt{n}.(\frac{n}{e})^n}{2^n} = \sqrt{2\pi} \lim_{n\to\infty} \sqrt{n}.(\frac{n}{2e})^n = \sqrt{2\pi}.\infty.\infty = \infty$$

Notações o e ω

- $o(g(n))$ é classe de funções $t(n)$ com **ordem de crescimento estritamente menor** que $g(n)$.

  Formalmente, $t(n) \in o(g(n))$ sss **para todo** $c > 0$, existe n_0 tq. $t(n) < c.g(n)$, $\forall n \geq n_0$.

- Usando limites:
- $t(n) \in o(g(n))$ **sss** $\lim_{n \to \infty} \frac{t(n)}{g(n)} = 0$

Notações o e ω

- $o(g(n))$ é classe de funções $t(n)$ com **ordem de crescimento estritamente menor** que $g(n)$.

  Formalmente, $t(n) \in o(g(n))$ sss **para todo** $c > 0$, existe n_0 tq. $t(n) < c.g(n)$, $\forall n \geq n_0$.

- Usando limites:
- $t(n) \in o(g(n))$ **sss** $\lim_{n \to \infty} \frac{t(n)}{g(n)} = 0$

Notações o e ω

- $o(g(n))$ é classe de funções $t(n)$ com **ordem de crescimento estritamente menor** que $g(n)$.

  Formalmente, $t(n) \in o(g(n))$ sss **para todo** $c > 0$, existe n_0 tq. $t(n) < c.g(n)$, $\forall n \geq n_0$.

- Usando limites:
- $t(n) \in o(g(n))$ **sss** $\lim_{n \to \infty} \frac{t(n)}{g(n)} = 0$

Notações o e ω

- $\omega(g(n))$ é classe de funções $t(n)$ com **ordem de crescimento estritamente maior** que $g(n)$.

  Formalmente, $t(n) \in \omega(g(n))$ sss para todo $c > 0$, existe n_0 tq. $t(n) > c.g(n)$, $\forall n \geq n_0$.

- Usando limites:
- $t(n) \in \omega(g(n))$ **sss** $\lim_{n \to \infty} \frac{t(n)}{g(n)} = \infty$

Notações o e ω

- $\omega(g(n))$ é classe de funções $t(n)$ com **ordem de crescimento estritamente maior** que $g(n)$.

  Formalmente, $t(n) \in \omega(g(n))$ sss para todo $c > 0$, existe n_0 tq. $t(n) > c.g(n)$, $\forall n \geq n_0$.

- Usando limites:
- $t(n) \in \omega(g(n))$ **sss** $\lim_{n \to \infty} \frac{t(n)}{g(n)} = \infty$

Propriedades das Notaçoes Assintóticas

Reflexividade

- $t(n) \in O(t(n))$
- $t(n) \in \Omega(t(n))$
- $t(n) \in \Theta(t(n))$

Transitividade

- Se $t(n) \in O(g(n))$ e $g(n) \in O(h(n))$, então $t(n) \in O(h(n))$
- Se $t(n) \in \Omega(g(n))$ e $g(n) \in \Omega(h(n))$, então $t(n) \in \Omega(h(n))$
- Se $t(n) \in \Theta(g(n))$ e $g(n) \in \Theta(h(n))$, então $t(n) \in \Theta(h(n))$

Propriedades das Notaçoes Assintóticas

Reflexividade

- $t(n) \in O(t(n))$
- $t(n) \in \Omega(t(n))$
- $t(n) \in \Theta(t(n))$

Transitividade

- Se $t(n) \in O(g(n))$ e $g(n) \in O(h(n))$, então $t(n) \in O(h(n))$
- Se $t(n) \in \Omega(g(n))$ e $g(n) \in \Omega(h(n))$, então $t(n) \in \Omega(h(n))$
- Se $t(n) \in \Theta(g(n))$ e $g(n) \in \Theta(h(n))$, então $t(n) \in \Theta(h(n))$

Propriedades das Notaçoes Assintóticas

Simetria

- $t(n) \in \Theta(g(n))$ se e somente se $g(n) \in \Theta(t(n))$

Simetria transposta

- $t(n) \in O(g(n))$ se e somente se $g(n) \in \Omega(t(n))$

Propriedades úteis

- $c \cdot t(n) \in O(t(n))$, para uma constante c
- Se $t_1(n) \in O(g_1(n))$ e $t_2(n) \in O(g_2(n))$, então $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

Teorema (Regra da Soma)

Se $t_1(n) \in O(g_1(n))$ e $t_2(n) \in O(g_2(n))$, então:

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

Prova

Já que $t_1(n) \in O(g_1(n))$ e $t_2(n) \in O(g_2(n))$, temos:

$$t_1(n) \leq c_1 \cdot g_1(n), \text{ para } n \geq n_1 \quad (1)$$

$$t_2(n) \leq c_2 \cdot g_2(n), \text{ para } n \geq n_2 \quad (2)$$

Se $c_3 = \max\{c_1, c_2\}$ e $n \geq \max\{n_1, n_2\}$, ao somar 1 e 2:

$$t_1(n) + t_2(n) \leq c_3[g_1(n) + g_2(n)]$$

$$t_1(n) + t_2(n) \leq 2 \cdot c_3 \max\{g_1(n), g_2(n)\}$$

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

Teorema (Regra da Soma)

Se $t_1(n) \in O(g_1(n))$ e $t_2(n) \in O(g_2(n))$, então:

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

Prova

Já que $t_1(n) \in O(g_1(n))$ e $t_2(n) \in O(g_2(n))$, temos:

$$t_1(n) \leq c_1 \cdot g_1(n), \text{ para } n \geq n_1 \quad (1)$$

$$t_2(n) \leq c_2 \cdot g_2(n), \text{ para } n \geq n_2 \quad (2)$$

Se $c_3 = \max\{c_1, c_2\}$ e $n \geq \max\{n_1, n_2\}$, ao somar 1 e 2:

$$t_1(n) + t_2(n) \leq c_3[g_1(n) + g_2(n)]$$

$$t_1(n) + t_2(n) \leq 2 \cdot c_3 \max\{g_1(n), g_2(n)\}$$

$$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$$

Propriedades das Notaçoes Assintóticas

Suponha que temos um algoritmo composta de k partes (procedimentos).

- A Regra da Soma implica que o **tempo do algoritmo** é determinada pela parte com **maior ordem de grandeza**, i.e., a parte mais demourada do algoritmo.

$$\left.\begin{array}{l} t_1(n) \in O(g_1(n)) \\ t_2(n) \in O(g_2(n)) \\ \quad\quad \vdots \\ t_k(n) \in O(g_k(n)) \end{array}\right\} \Rightarrow T(n) = t_1(n) + t_2(n) + \cdots + t_k(n)$$

$$T(n) \in O(\max\{g_1(n), g_2(n), \ldots, g_k(n)\})$$

Teorema (Regra do Produto)

Se $t_1(n) \in O(g_1(n))$ e $t_2(n) \in O(g_2(n))$, então:

$$t_1(n).t_2(n) \in O(g_1(n).g_2(n))$$

Ordens de grandeza de funções importantes

- Todas as **funções logarítmicas** $\log_a n$ pertencem a mesma classe $\Theta(\log n)$, independente da base $a > 1$ do logaritmo.

Ordens de grandeza de funções importantes

- Todas as **funções logarítmicas** $\log_a n$ pertencem a mesma classe $\Theta(\log n)$, independente da base $a > 1$ do logaritmo.
- Todos os **polinômios de grau** k pertencem à mesma classe: $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 \in \Theta(n^k)$.

Ordens de grandeza de funções importantes

- Todas as **funções logarítmicas** $\log_a n$ pertencem a mesma classe $\Theta(\log n)$, independente da base $a > 1$ do logaritmo.
- Todos os **polinômios de grau** k pertencem à mesma classe: $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 \in \Theta(n^k)$.
- **Funções exponenciais** a^n possuem diferentes ordens de grandeza para diferentes bases a.
  Ex.: $2^n \in O(2^n), 3^n \in O(3^n)$

Ordens de grandeza de funções importantes

- Todas as **funções logarítmicas** $\log_a n$ pertencem a mesma classe $\Theta(\log n)$, independente da base $a > 1$ do logaritmo.
- Todos os **polinômios de grau** k pertencem à mesma classe: $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 \in \Theta(n^k)$.
- **Funções exponenciais** a^n possuem diferentes ordens de grandeza para diferentes bases a.
  Ex.: $2^n \in O(2^n), 3^n \in O(3^n)$
- $\log n < n^k < a^n < n! < n^n$.
  (considerando $k > 0, a > 1$)

Ordens de grandeza de funções importantes

- Todas as **funções logarítmicas** $\log_a n$ pertencem a mesma classe $\Theta(\log n)$, independente da base $a > 1$ do logaritmo.
- Todos os **polinômios de grau** k pertencem à mesma classe: $a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 \in \Theta(n^k)$.
- **Funções exponenciais** a^n possuem diferentes ordens de grandeza para diferentes bases a.
  Ex.: $2^n \in O(2^n), 3^n \in O(3^n)$
- $\log n < n^k < a^n < n! < n^n$.
  (considerando $k > 0, a > 1$)
  Se $f(n) = c$, uma constante não negativa, então $f(n) \in \Theta(1)$

Classes básicas de crescimento assintótico

Classe	Nome
1	constante
$\log n$	logarítmica
n	linear
$n \log n$	linearítmica
n^2	quadrática
n^3	cúbica
2^n	exponencial
$n!$	fatorial

Notações Assintóticas na Análise de Algoritmos

O que significa dizer que o tempo de execução de um algoritmo é $O(g(n))$?

- Significa que, o tempo de execução do algoritmo no pior caso é $O(g(n))$.
- Ou seja, o tempo de execução do algoritmo é **no máximo uma constante vezes g(n)** (i.e. $c.g(n)$) para qualquer entrada de tamanho n suficientemente grande.
- A notação O é usado para descrever o **limite superior** do **tempo de execução** de um algoritmo.

Notações Assintóticas na Análise de Algoritmos

O que significa dizer que o tempo de execução de um algoritmo é $O(g(n))$?

- Significa que, o tempo de execução do algoritmo no pior caso é $O(g(n))$.
- Ou seja, o tempo de execução do algoritmo é **no máximo uma constante vezes g(n)** (i.e. $c.g(n)$) para qualquer entrada de tamanho n suficientemente grande.
- A notação O é usado para descrever o **limite superior** do **tempo de execução** de um algoritmo.

Notações Assintóticas na Análise de Algoritmos

Exemplo: o tempo de execução no **pior caso** do algoritmo **insertion-sort** é: $T(n) = \frac{1}{2}n(n-1)$.

- Ou seja, **para qualquer entrada**, o tempo de execução do algoritmo é $O(n^2)$
- Ou seja, o algoritmo nunca gasta mais do que $c.n^2$ (para algum $c > 0$), para todo n suficiente mente grande.
- $c.n^2$ é um **limite superior** para o tempo de execução do algoritmo insertion-sort.

Notações Assintóticas na Análise de Algoritmos

O que significa dizer que o tempo de execução de um algoritmo é $\Omega(g(n))$?

- Significa que, o tempo de execução do algoritmo no melhor caso é $\Omega(g(n))$.
- Ou seja, o tempo de execução do algoritmo é pelo menos uma constante vezes $g(n)$ (i.e. $c.g(n)$), para qualquer entrada de tamanho n suficientemente grande.
- A notação Ω é usada para descrever o **limite inferior** do **tempo de execução** para todas as entradas do algoritmo.

Notações Assintóticas na Análise de Algoritmos

O que significa dizer que o tempo de execução de um algoritmo é $\Omega(g(n))$?

- Significa que, o tempo de execução do algoritmo no melhor caso é $\Omega(g(n))$.
- Ou seja, o tempo de execução do algoritmo é pelo menos uma constante vezes $g(n)$ (i.e. $c.g(n)$), para qualquer entrada de tamanho n suficientemente grande.
- A notação Ω é usada para descrever o **limite inferior** do **tempo de execução** para todas as entradas do algoritmo.

Notações Assintóticas na Análise de Algoritmos

Exemplo: o tempo de execução no **melhor caso** do algoritmo **insertion-sort** é: $T(n) = n - 1$.

- Ou seja, para qualquer entrada, o tempo de execução do algoritmo insertion-sort é $\Omega(n)$
- Ou seja, o tempo do algoritmo é **pelo menos** $c.n$ (para algum $c > 0$), para todo n suficiente mente grande.
- $c.n$ é um **limite inferior** para o tempo de execução do algoritmo insertion-sort.

$\Rightarrow$ O tempo de execução do algoritmo Insertion-Sort está entre $\Omega(n)$ e $O(n^2)$.

Notações Assintóticas na Análise de Algoritmos

Exemplo: o tempo de execução no **melhor caso** do algoritmo **insertion-sort** é: $T(n) = n - 1$.

- Ou seja, para qualquer entrada, o tempo de execução do algoritmo insertion-sort é $\Omega(n)$
- Ou seja, o tempo do algoritmo é **pelo menos** $c.n$ (para algum $c > 0$), para todo n suficiente mente grande.
- $c.n$ é um **limite inferior** para o tempo de execução do algoritmo insertion-sort.

$\Rightarrow$ O tempo de execução do algoritmo Insertion-Sort está entre $\Omega(n)$ e $O(n^2)$.

Notações Assintóticas na Análise de Algoritmos

Algoritmo Ótimo

- Se um algoritmo possui o mesmo tempo de execução no melhor caso e pior caso, então o algoritmo possui tempo ótimo ou justo.
- O tempo de um **algoritmo ótimo** é descrito usando a notação Θ.

Exemplo:

Um algoritmo para calcular o fatorial de um número n tem tempo de execução $\Theta(n)$.

Para o algoritmo Insertion-Sort, podemos afirmar que o tempo de execução **no pior caso** é $\Theta(n^2)$.

No entanto, se quisermos cobrir **todos os casos**, a afirmação mais estrita seria que o Insertion-Sort executa em tempo $O(n^2)$.

Notações Assintóticas na Análise de Algoritmos

Algoritmo Ótimo

- Se um algoritmo possui o mesmo tempo de execução no melhor caso e pior caso, então o algoritmo possui tempo ótimo ou justo.
- O tempo de um **algoritmo ótimo** é descrito usando a notação Θ.

Exemplo:

Um algoritmo para calcular o fatorial de um número n tem tempo de execução $\Theta(n)$.

Para o algoritmo Insertion-Sort, podemos afirmar que o tempo de execução **no pior caso** é $\Theta(n^2)$.

No entanto, se quisermos cobrir **todos os casos**, a afirmação mais estrita seria que o Insertion-Sort executa em tempo $O(n^2)$.

Notações Assintóticas na Análise de Algoritmos

Algoritmo Ótimo

- Se um algoritmo possui o mesmo tempo de execução no melhor caso e pior caso, então o algoritmo possui tempo ótimo ou justo.
- O tempo de um **algoritmo ótimo** é descrito usando a notação Θ.

Exemplo:

Um algoritmo para calcular o fatorial de um número n tem tempo de execução $\Theta(n)$.

Para o algoritmo Insertion-Sort, podemos afirmar que o tempo de execução **no pior caso** é $\Theta(n^2)$.

No entanto, se quisermos cobrir **todos os casos**, a afirmação mais estrita seria que o Insertion-Sort executa em tempo $O(n^2)$.