UFRR

UNIVERSIDADE FEDERAL DE RORAIMA

Comportamento Assintótico

Aula Baseada nos slides da Profª. Dr. Rosiane Rodrigues e Profº. Eduardo Nakamura- UFAM

Prof. Dr. Herbert Oliveira Rocha
herberthb12@gmail.com

Algoritmos

Aplicações práticas de algoritmos:

- ✓ O projeto Genoma Humano – Identificação de genes do DNA humano
- ✓ A internet – Conexão entre pessoas
- ✓ Comércio eletrônico – Segurança (Teoria do números)
- ✓ Na indústria – Alocação de recursos
- ✓ Rotas e distâncias

Algoritmos

Imagine que os computadores fossem infinitamente rápidos e que a memória fosse livre

- ✓ **Você teria alguma razão para estudar algoritmos?**

Algoritmos

Imagine que os computadores fossem infinitamente rápidos e que a memória fosse livre

- ✓ **Você teria alguma razão para estudar algoritmos?**

- ✓ SIM para demonstrar que a sua solução termina e o faz com a resposta correta

Na REALIDADE

- ✓ Computadores podem ser rápidos, mas não são infinitamente rápidos
- ✓ A memória pode ser de baixo custo, mas não é gratuita

Algoritmos

Analisar problemas e algoritmos do ponto de vista computacional, significa determinar como os mesmos se **comportam** para pequenas e, principalmente, **grandes instâncias** e para as casos mais comuns, os melhores casos e, principalmente, os **casos mais difíceis** de ocorrerem.

Algoritmos

INSTÂNCIA DE UM PROBLEMA

É obtida pela fixação de valores particulares de todos os parâmetros do problema.

Conceitos

Algoritmos

EXEMPLO:

Ordenar uma lista finita de *n* números inteiros.

PARÂMETROS: ***n*** **números inteiros** $(x_1, x_2, \ldots, x_n)$

INSTÂNCIA: **(2, 31, 0, -7),** ***n*****=4**

SOLUÇÃO: **os *n* inteiros em ordem crescente.**

(-7, 0, 2, 31)

Conceitos

Resolução de Problemas por Computador

Dar uma solução para um problema computacional significa elaborar um **ALGORITMO** em **PSEUDO-CÓDIGO** (linguagem intermediária entre a linguagem natural e a linguagem de programação) e implementá-lo numa linguagem de programação, gerando um **PROGRAMA**.

CUSTO DE UTILIZAÇÃO DE UM ALGORITMO

0 custo de um algoritmo pode ser medido de várias formas:

- ✓ Através da **execução do programa** em um computador real (tempo de execução medido diretamente).
- ✓ Através do uso de um **modelo matemático**.

Algoritmos

ESTRUTURA GERAL DE UM ALGORITMO

- **Algoritmo *(Nome-do-Algoritmo);***
- **Declaração de constantes, tipos e variáveis;**
- **Início**
  - **Atribuições;**
  - **Sequências de Comandos;**
  - **Estruturas de controle:**
  - **Seleção**
  - **Repetição**
  - **Comandos de E/S.**
- **Fim.**

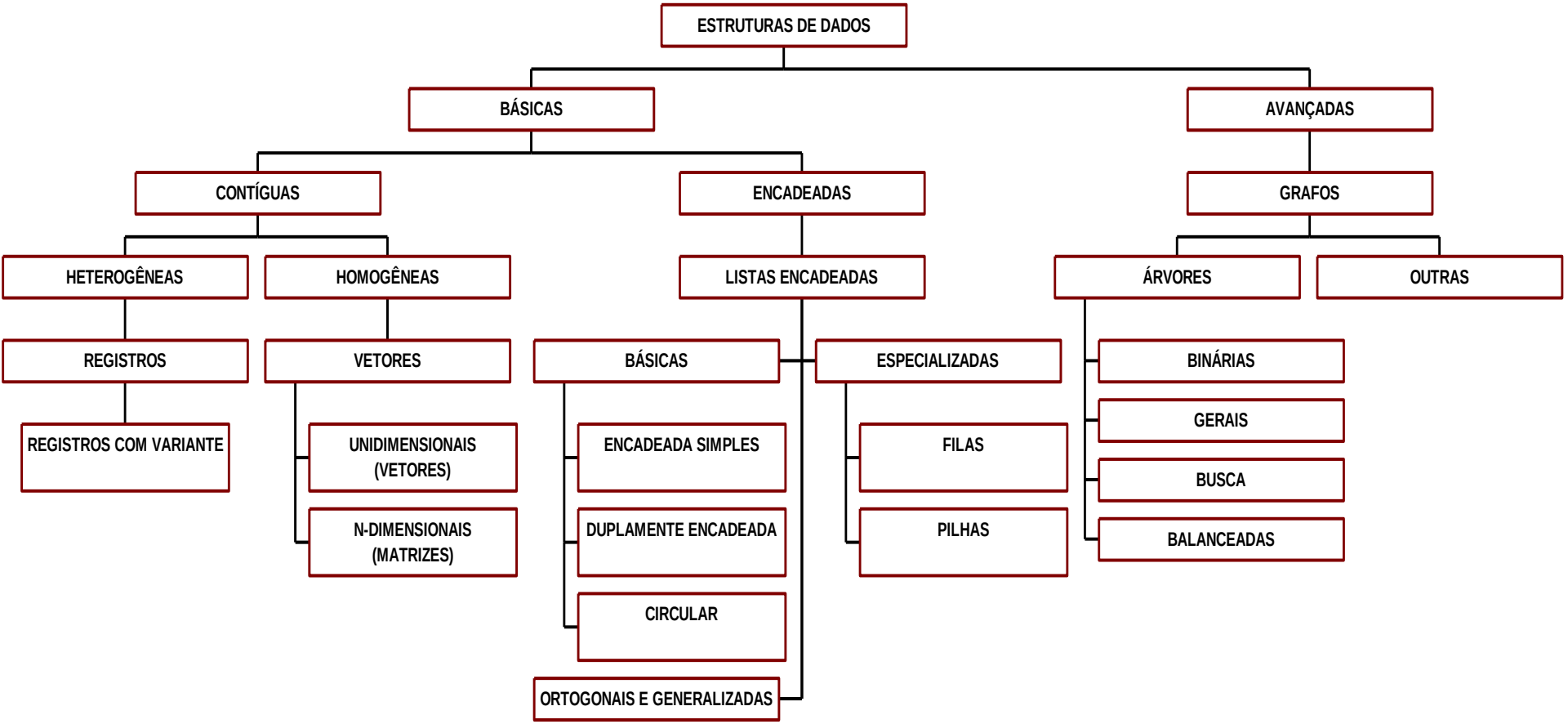
Estruturas de Dados

ESTRUTURAS DE DADOS

- São formas de organização da informação, com o objetivo de manipulá-la (inserir, retirar, alterar, consultar) mais eficientemente.

Conceitos

Esquema Geral de Estruturas de Dados



Análise Assintótica

Ordem de crescimento

- ✓ Caracterização simples da eficiência do algoritmo
- ✓ Comparar desempenho relativo de algoritmos alternativos

Quando observamos a ordem de crescimento

- ✓ Estamos estudando a eficiência **assintótica** dos algoritmos
- ✓ Estamos preocupados como o tempo de execução aumenta com tamanho da entrada

Análise Assintótica

A notação assintótica

- ✓ Usada para descrever o tempo de execução assintótica de um algoritmo
- ✓ Facilita a análise para valores de entrada grandes

Na análise

- ✓ Comportamento a ser observado em uma função **f(n)**, quando **n** tende ao infinito (análise do crescimento assintótico da função).
- ✓ O custo assintótico de uma função **f(n)** representa o limite do comportamento de custo quando **n** cresce.
- ✓ Em geral, o custo aumenta com o tamanho **n** do problema.

Análise Assintótica

IMPORTANTE

Para **valores pequenos de n**, mesmo um algoritmo ineficiente não custa muito para ser executado.

Análise Assintótica

NOTAÇÕES

Análise Assintótica

O que significa $T(n) = \Theta(n^2)$?

DOMINAÇÃO ASSINTÓTICA

Para uma dada função $g(n)$, denotamos por $\Theta(g(n))$ o conjunto de funções

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0, \text{ tais que } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \; \forall \; n > n_0\}$$

Análise Assintótica

NOTAÇÃO Θ (teta)

- f(n) = Θ(g(n))

  - Lê-se f(n) é da mesma ordem de g(n).
  - Θ limita a função por fatores constantes.

- g(n) é um limite assintótico superior e inferior para f(n)

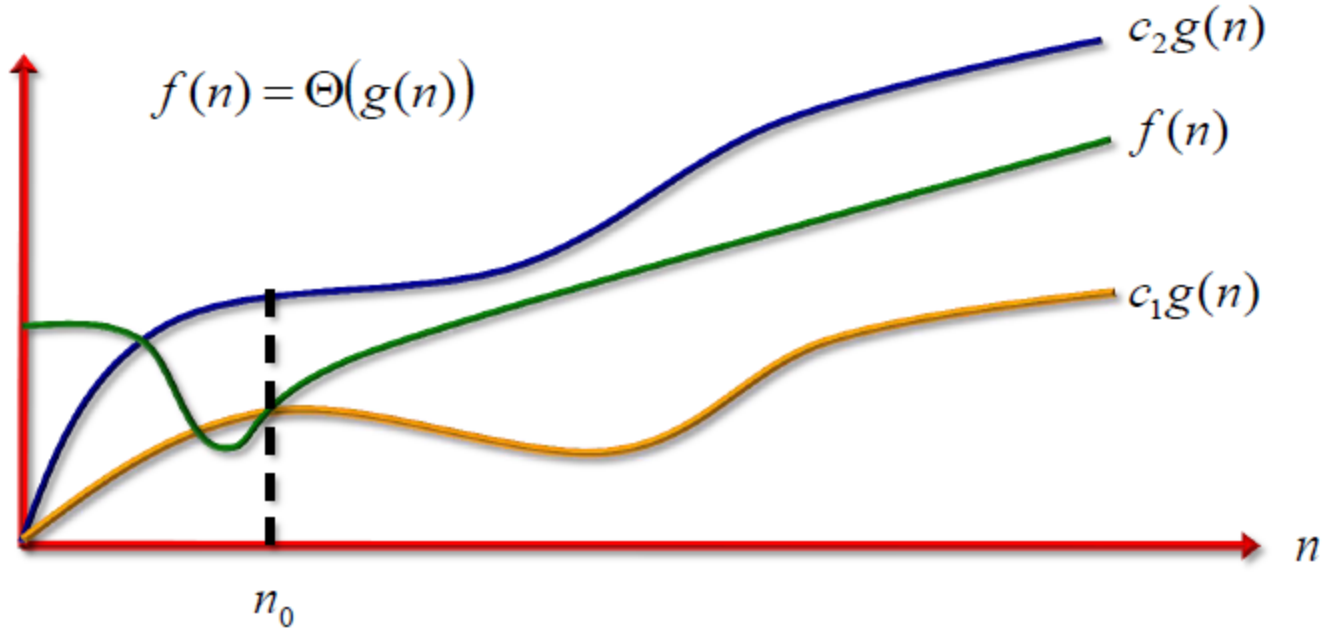
ou

- g(n) é um limite assintótico firme para f(n).

Análise Assintótica

Notação Θ

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0, \text{ tais que } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \ \forall \ n > n_0\}$$



Análise Assintótica

Mostre que se $T(n) = \frac{1}{2}n^2 - 4n$, então $T(n) = \Theta(n^2)$

Aplicando a definição de Θ temos:

$$0 \leq c_1 n^2 \leq \frac{1}{2}n^2 - 4n \leq c_2 n^2$$

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0, \text{ tais que } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \; \forall \; n > n_0\}$$

Análise Assintótica

Mostre que se $T(n) = \frac{1}{2}n^2 - 4n$, então $T(n) = \Theta(n^2)$

Escolhendo $n_0 = 16$, obtemos:

DICA: Dividir por 16^2

$$0 \leq c_1 \leq \frac{1}{2} - \frac{4}{16} \leq c_2 \qquad \Rightarrow \qquad 0 \leq c_1 \leq \frac{1}{4} \leq c_2$$

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0, \text{ tais que } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \; \forall \; n > n_0\}$$

Análise Assintótica

Mostre que se $T(n) = \frac{1}{2}n^2 - 4n$, então $T(n) = \Theta(n^2)$

Escolhendo $c_1 = \frac{1}{5}$ e $c_2 = \frac{1}{2}$, obtemos:

$$0 \le c_1 \le \frac{1}{4} \le c_2 \qquad \Rightarrow \qquad 0 \le \frac{1}{5} \le \frac{1}{4} \le \frac{1}{2}$$

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0, \text{ tais que } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \ \forall \ n > n_0\}$$

Análise Assintótica

Mostre que se $T(n) = \frac{1}{2}n^2 - 4n$, então $T(n) = \Theta(n^2)$

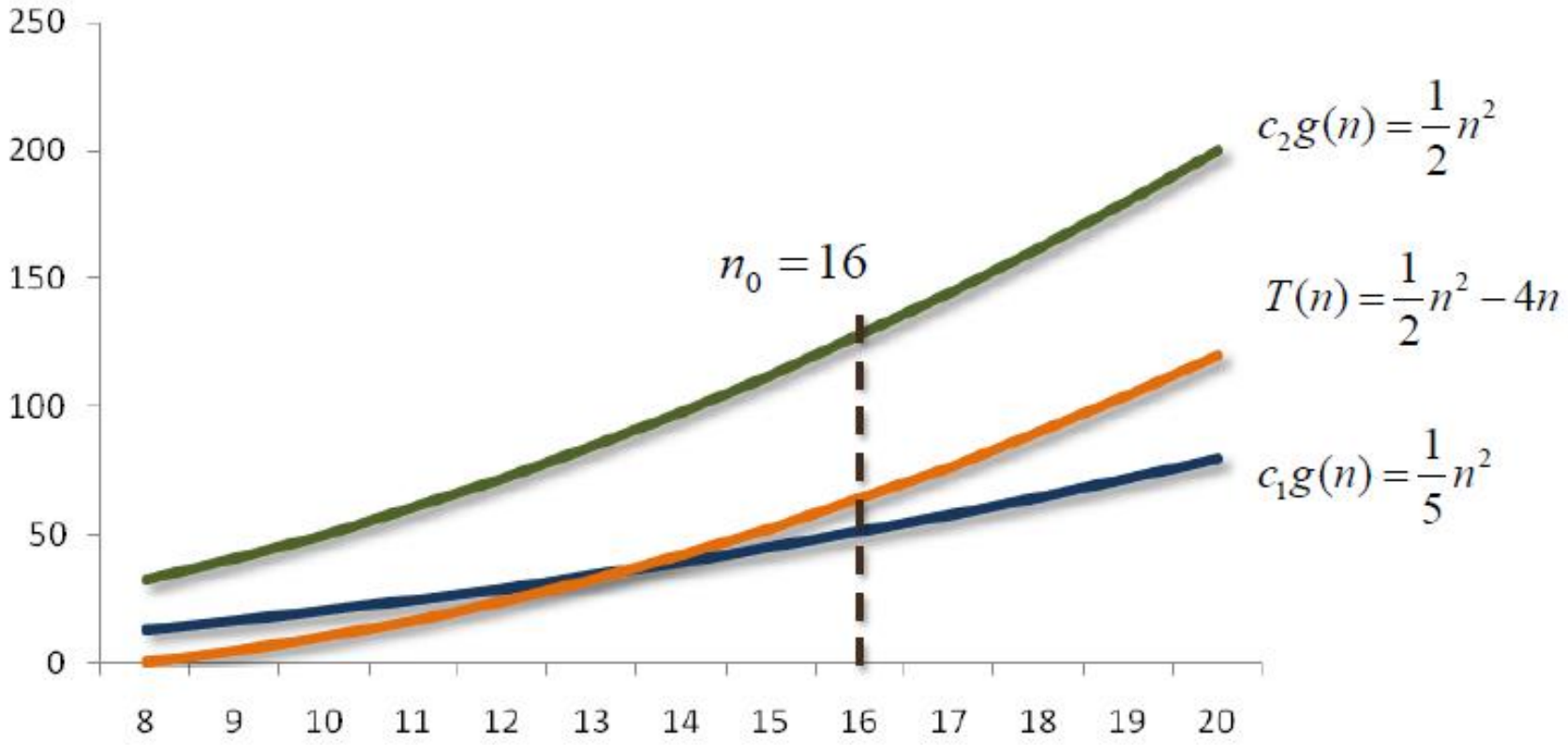
Portanto, para $n = 16$, temos que

$$0 \leq \frac{1}{5}n^2 \leq \frac{1}{2}n^2 - 4n \leq \frac{1}{2}n^2$$

$$\Theta(g(n)) = \{f(n) : \exists \text{ constantes positivas } c_1, c_2 \text{ e } n_0, \text{ tais que } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \; \forall \; n > n_0\}$$

Análise Assintótica

Graficamente, temos:



Análise Assintótica

Notação O

- ✓ Quando temos apenas um **limite assintótico superior**, usamos a notação *O*
- ✓ Usamos a notação *O* para darmos um limite superior

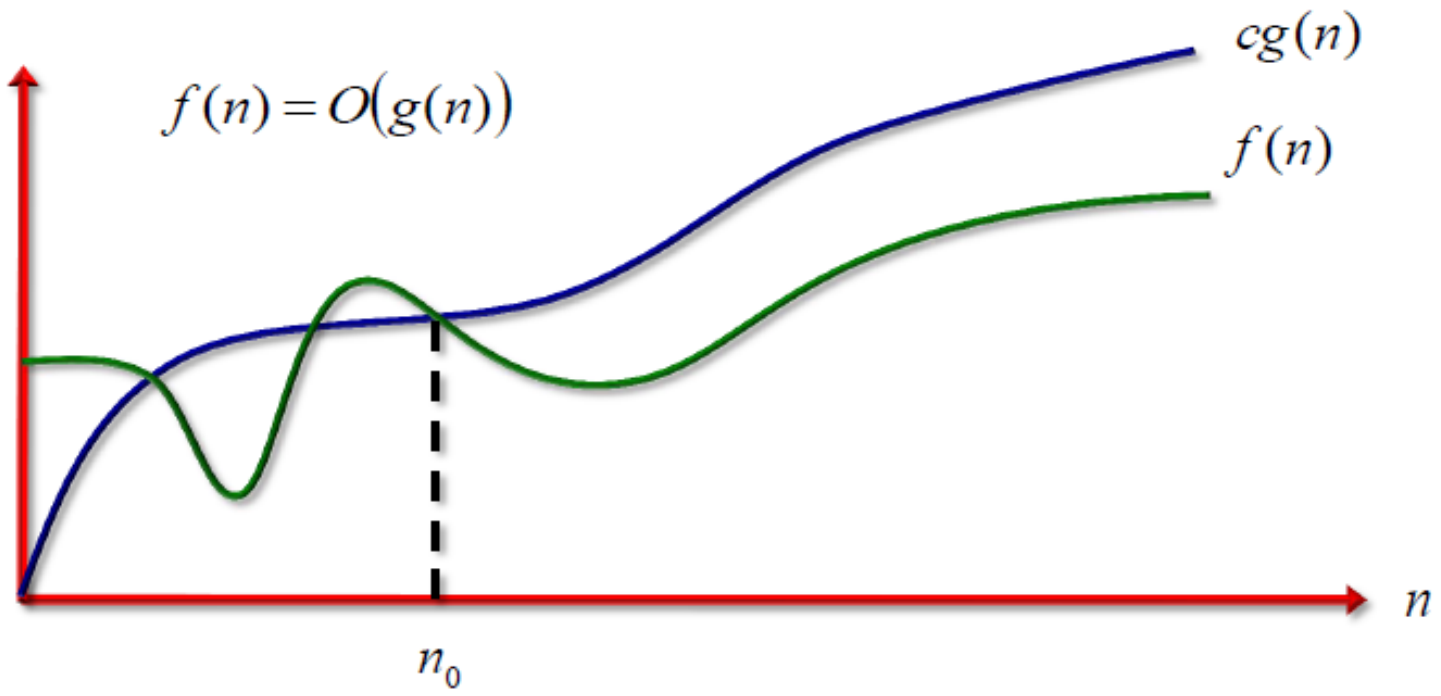
A definição formal é

$$O(g(n)) = \{f(n) : \exists \text{ constantes positivas } c \text{ e } n_0, \text{ tais que } 0 \leq f(n) \leq cg(n) \; \forall \; n > n_0\}$$

Análise Assintótica

Notação O

$$O(g(n)) = \{f(n) : \exists \text{ constantes positivas } c \text{ e } n_0, \text{ tais que } 0 \leq f(n) \leq cg(n) \; \forall \; n > n_0\}$$



Análise Assintótica

NOTAÇÃO O

A **notação O** é usada para expressar o limite superior do tempo de execução de cada algoritmo para resolver um problema específico (limite superior de cada algoritmo para um problema).

NOTAÇÃO O

REGRA DA SOMA $O(f(n)) + O(g(n))$

- Suponha 3 trechos de programas cujos tempos de execução sejam $O(n)$, $O(n^2)$ e $O(n\log n)$.
- O tempo de execução dos 2 primeiros trechos é $O(\max(n,n^2))$, que é $O(n^2)$.
- O tempo de execução de todos os 3 trechos é, então, $O(\max(n^2, n\log n))$, que é $O(n^2)$

NOTAÇÃO O

Exemplo:

$f(n) = (n+1)^2$ e $g(n) = 2n^2$, $f(n) = O(n^2)$?

$$O(g(n)) = \{f(n) : \exists \text{ constantes positivas } c \text{ e } n_0, \text{ tais que } 0 \leq f(n) \leq cg(n) \; \forall \; n > n_0\}$$

NOTAÇÃO O

Exemplo:

Assim $f(n) = O(g(n^2))$ - Porquê ?

$(n + 1)^2 \leq cn^2$

$n^2 + 2n + 1 \leq cn^2$

$2n + 1 \leq (c - 1)n^2$

n	(n+1)^2	2n^2
0	1	0
1	4	2
2	9	8
3	16	18
4	25	32

Análise Assintótica

NOTAÇÃO Ω

(*ômega grande*)

$$\mathbf{f(n) = \Omega(g(n))}$$

- Lê-se **f(n)** é de ordem no mínimo **g(n)**
- Ω define um limite inferior para a função, por um fator constante.

f (n) *é* um limite assintótico inferior para **g(n)**

NOTAÇÃO Ω

Exemplo:

O limite inferior para qualquer algoritmo de ordenação que utiliza comparação entre elementos é $\Omega(n \log n)$.

Análise Assintótica

Notação Ω

- ✓ Quando temos apenas um **limite assintótico inferior**, usamos a notação **Ω**
- ✓ Usamos a notação **Ω** para darmos um limite inferior

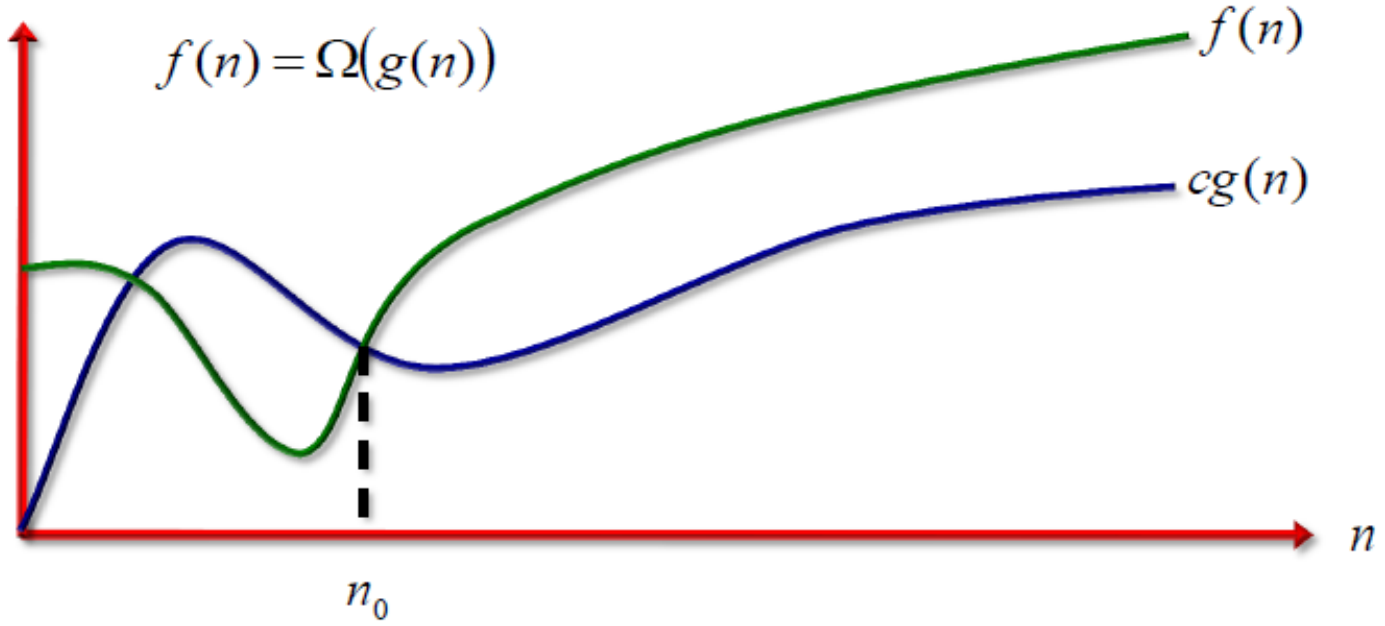
A definição formal é

$$\Omega(g(n)) = \{f(n) : \exists \text{ constantes positivas } c \text{ e } n_0, \text{ tais que } 0 \leq cg(n) \leq f(n) \; \forall \; n > n_0\}$$

Análise Assintótica

Notação Ω

$$\Omega(g(n)) = \{f(n) : \exists \text{ constantes positivas } c \text{ e } n_0, \text{ tais que } 0 \le cg(n) \le f(n) \ \forall \ n > n_0\}$$



Análise Assintótica

TEOREMA

Para duas funções quais $f(n)$ e $g(n)$, temos que $f(n) = \Theta(g(n))$ se e somente se, $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$.

Comportamento Assintótica

NOTAÇÃO o
(little O ou "o" pequeno)

O limite assintótico superior definido pela notação O pode ser assintoticamente firme ou não.

EXEMPLO:

O limite $2n^2 = O(n^2)$ é assintoticamente firme

O limite $2n = O(n^2)$ não é assintoticamente firme.

A notação **o** é usada para definir **um limite superior que não é assintoticamente firme**.

Comportamento Assintótica

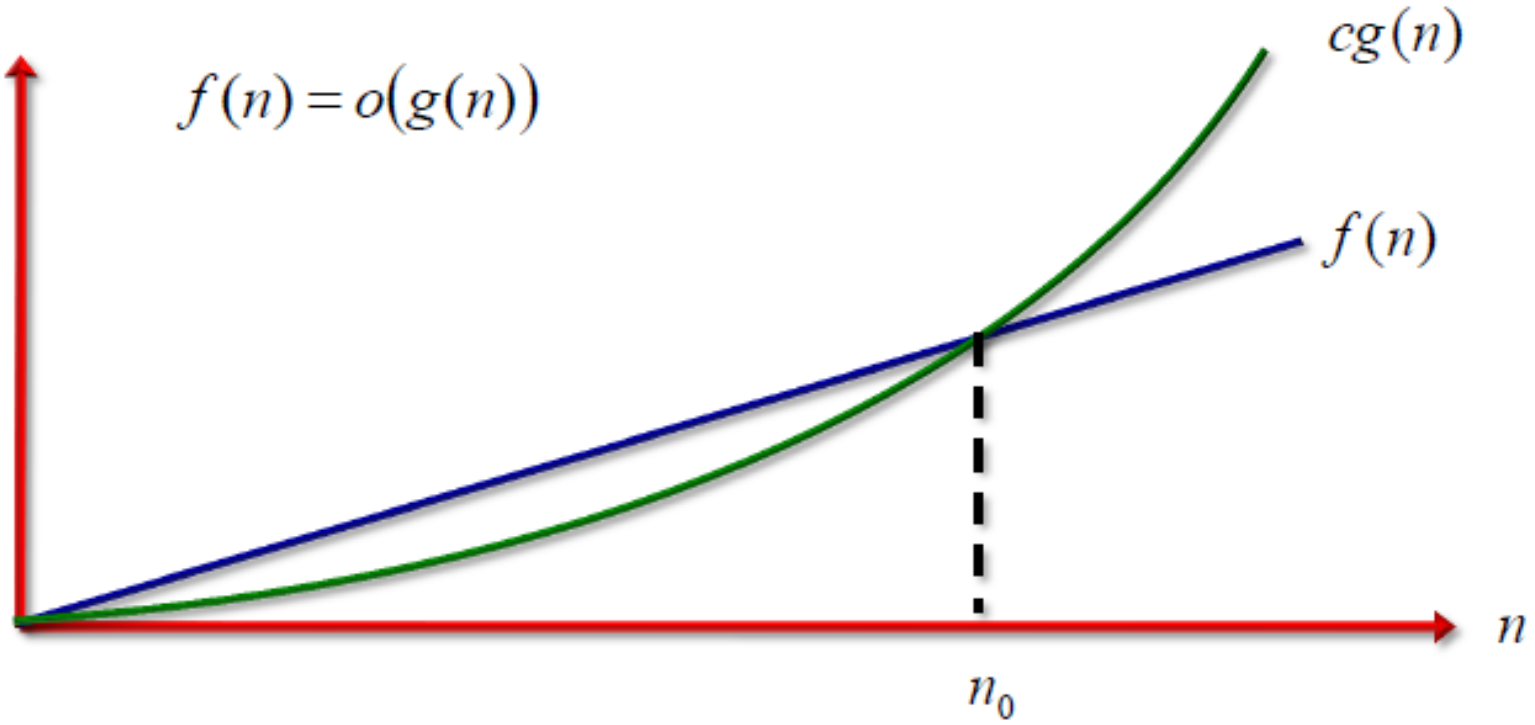
NOTAÇÃO o
(little O ou "o" pequeno)

Formalmente temos:

$$o(g(n)) = \{f(n) : \forall \text{ constante } c > 0, \exists \text{ uma constante } n_0 > 0,$$
$$\text{tal que } 0 \leq f(n) < cg(n) \ \forall \ n > n_0\}$$

Comportamento Assintótica

$$o(g(n)) = \{f(n) : \forall \text{ constante } c > 0, \exists \text{ uma constante } n_0 > 0, \text{ tal que } 0 \leq f(n) < cg(n) \ \forall \ n > n_0\}$$



Comportamento Assintótica

$$o(g(n)) = \{f(n) : \forall \text{ constante } c > 0, \exists \text{ uma constante } n_0 > 0, \text{ tal que } 0 \leq f(n) < cg(n) \; \forall \; n > n_0\}$$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Em matemática, o conceito de limite é usado para descrever o comportamento de uma função à medida que o seu argumento se aproxima de um determinado valor

NOTAÇÃO ω

(*ômega pequeno*)

O limite assintótico inferior definido pela notação **ω** pode ser assintoticamente firme ou não.

EXEMPLO:

O limite $2n^2 = \omega(n^2)$ é assintoticamente firme

O limite $2n^2 = \omega(n)$ não é assintoticamente firme.

A notação **ω** é usada para definir um limite inferior que não é assintoticamente firme.

Comportamento Assintótica

NOTAÇÃO ω

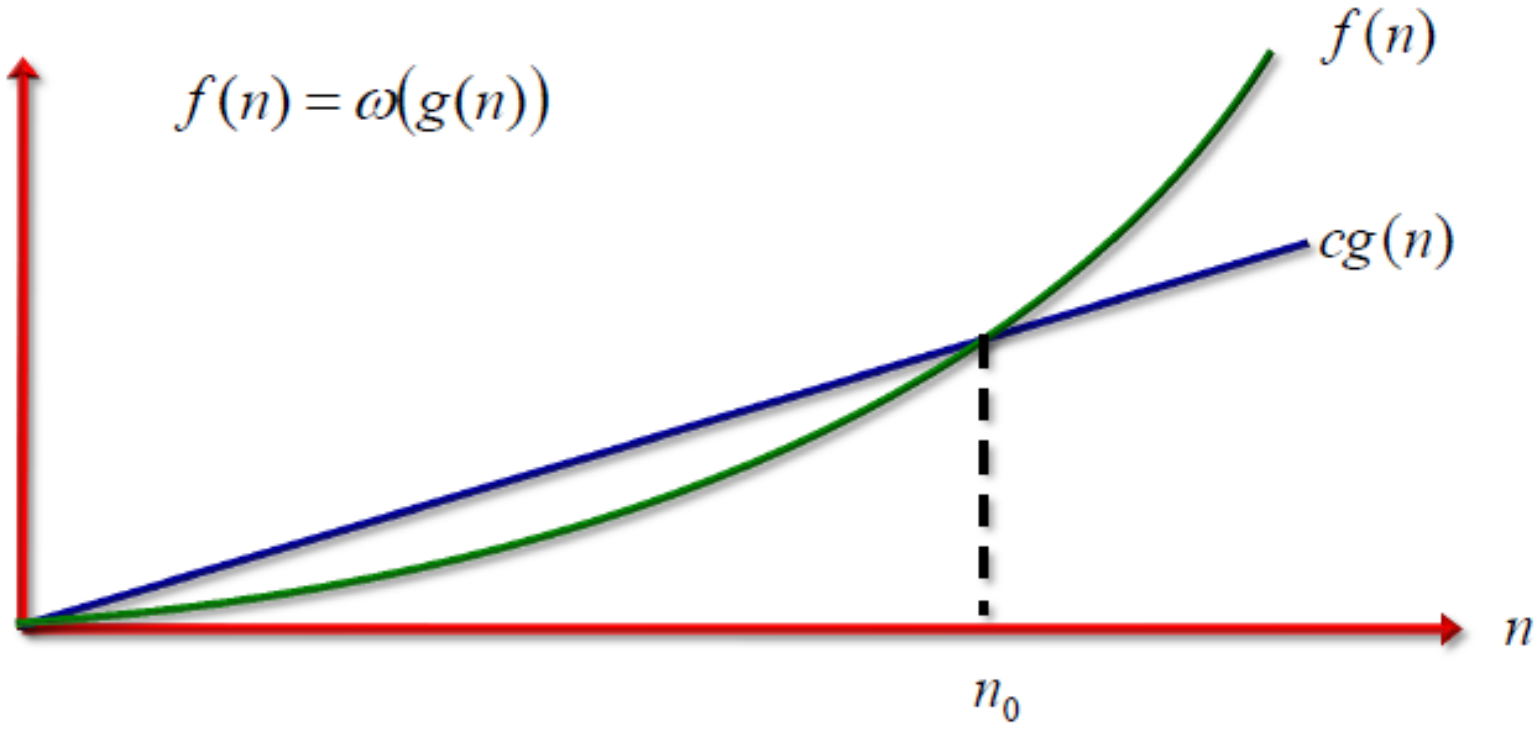
(*ômega pequeno*)

Formalmente temos:

$$\omega(g(n)) = \{f(n) : \forall \text{ constante } c > 0, \exists \text{ uma constante } n_0 > 0, \text{ tal que } 0 \leq cg(n) < f(n) \ \forall \ n > n_0\}$$

Comportamento Assintótica

$$\omega(g(n)) = \{f(n) : \forall \text{ constante } c > 0, \exists \text{ uma constante } n_0 > 0, \text{ tal que } 0 \leq cg(n) < f(n) \ \forall \ n > n_0\}$$



Comportamento Assintótica

$$\omega(g(n)) = \{f(n) : \forall \text{ constante } c > 0, \exists \text{ uma constante } n_0 > 0, \text{ tal que } 0 \leq cg(n) < f(n) \; \forall \; n > n_0\}$$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

Comportamento Assintótica

Usando as definições de $\Theta, O, \Omega, o \; e \; \omega$, mostre que:

a) $n \log n + 5n = \Theta(n \log n)$

b) $2^{n+1} = O(2^n)$

c) $2n^2 - 1 = \Omega(n^2)$

d) $n^2 = o(n^3)$

e) $n^2 = \omega(n)$

f) $n \log n = o(n^2)$

g) $2n^2 \neq o(n^2)$

Comportamento Assintótica

Classes de Comportamento Assintótico

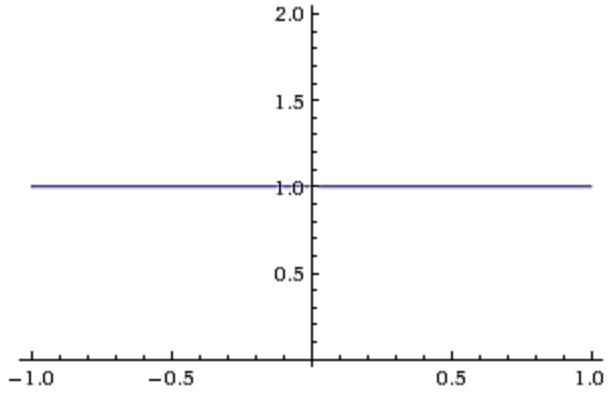
- Se **f** é uma função de complexidade para um **algoritmo F**, então **O(f)** é considerada a complexidade assintótica, ou o comportamento assintótico do algoritmo **F**.

- A relação de dominação assintótica permite **comparar funções de complexidade**

Comportamento Assintótica

Classes de Comportamento Assintótico

***f (n)* = *O*(1) (complexidade constante)**

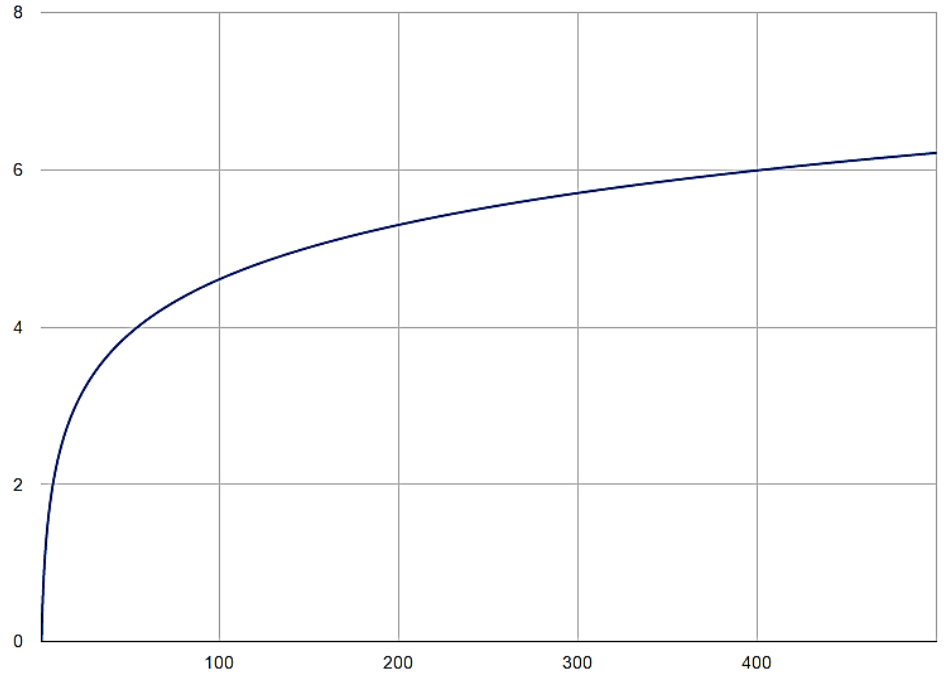


- ✓ O uso do algoritmo independe do tamanho de **n**.
- ✓ Neste caso as instruções do algoritmo são executadas um número fixo de vezes.

Comportamento Assintótica

f(*n*) = *O*(log n) (complexidade logarítmica)

- ✓ Ocorre tipicamente em algoritmos que resolvem um problema transformando-o em problemas menores.



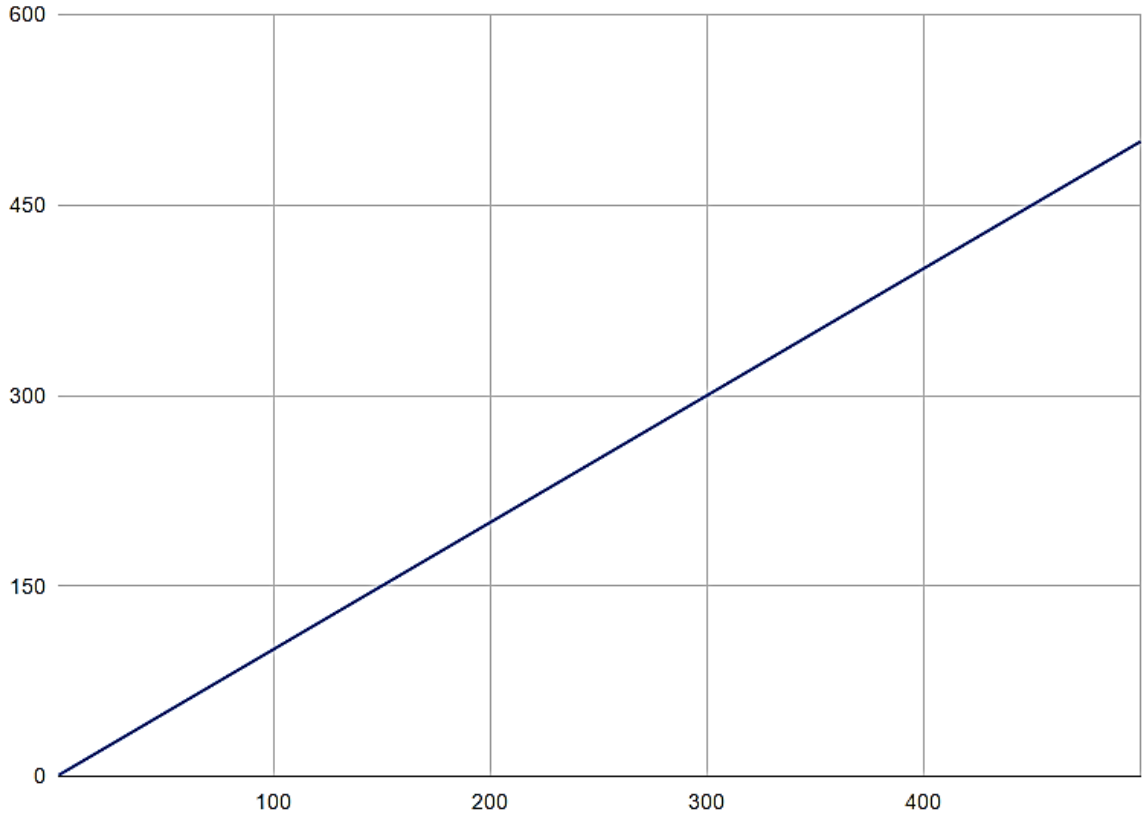
Comportamento Assintótica

***f(n)= O(n)* (complexidade de linear)**

- ✓ Em geral um pequeno trabalho é realizado sobre cada elemento de entrada.
- ✓ Esta é a melhor situação possível para um algoritmo que tem que processar n elementos de entrada ou produzir n elementos de saída.
- ✓ Cada vez que n dobra de tamanho o tempo de execução dobra.

Comportamento Assintótica

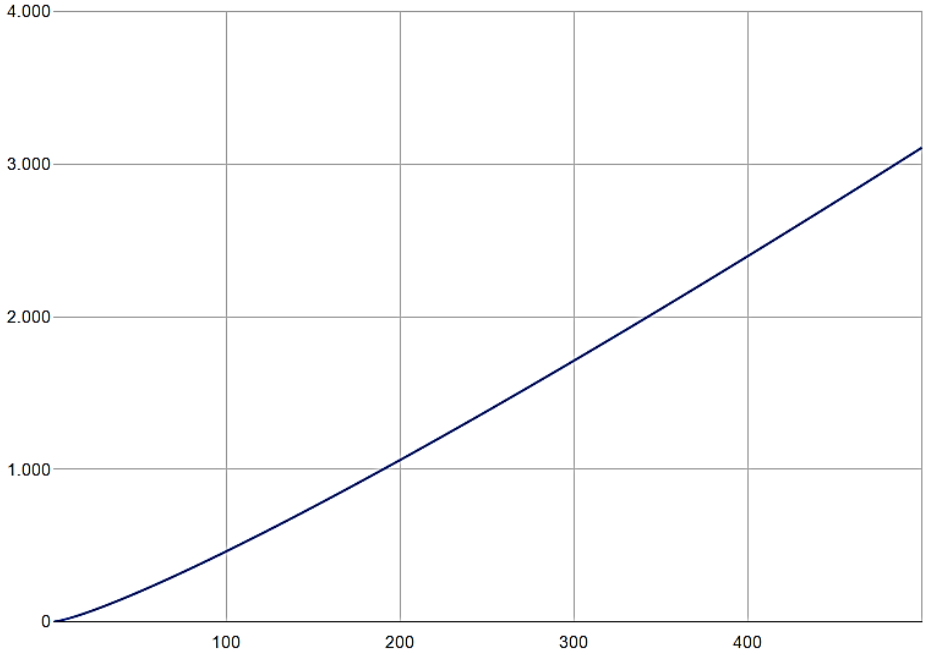
***f(n)= O(n)* (complexidade de linear)**



Comportamento Assintótica

f(n)= O(nlogn)

- ✓ Este tempo de execução ocorre tipicamente em algoritmos que resolvem um problema quebrando-o em problemas menores, resolvendo cada um deles independentemente e depois ajuntando as soluções.



Comportamento Assintótica

f(n)= O(n^2) (complexidade quadrática)

- ✓ Algoritmos desta ordem de complexidade ocorrem quando os itens de dados são processados aos pares, muitas vezes em um anel (*loop*) dentro de outro.

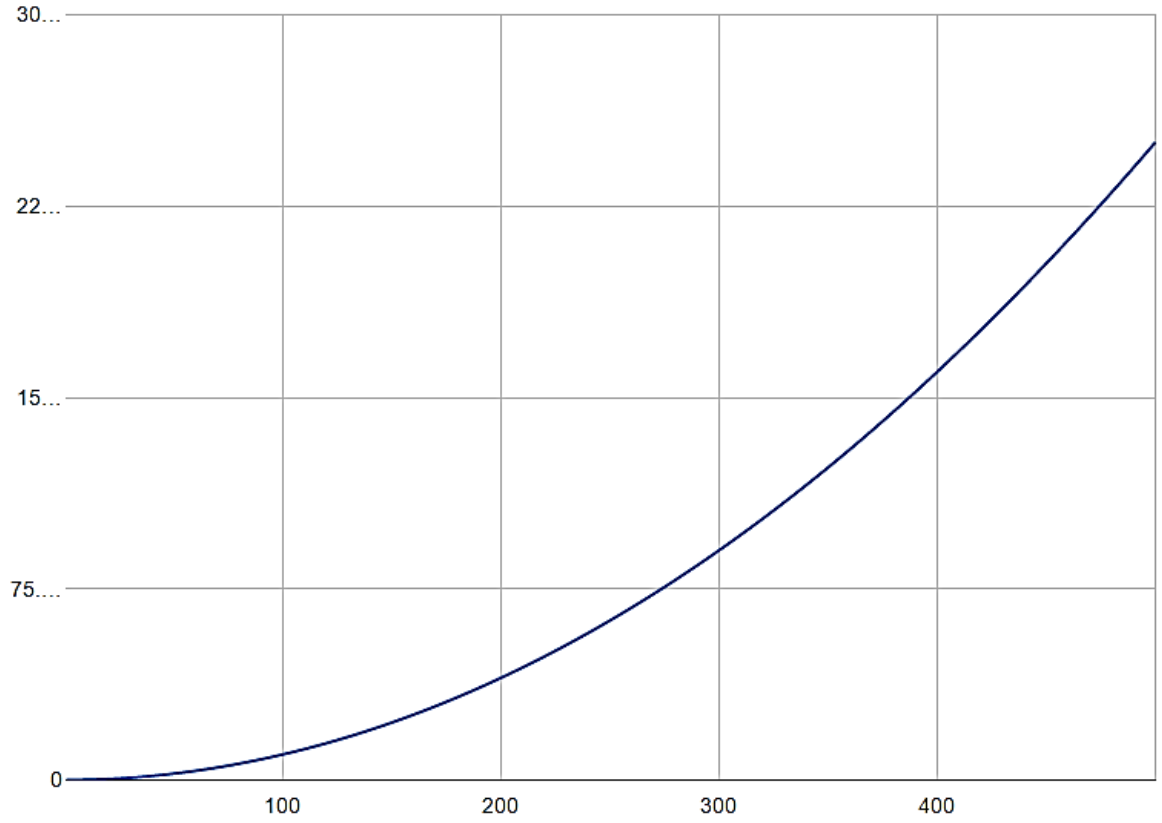
Exemplo:

Quando n é mil, o número de operações é da ordem de 1 milhão.

- ✓ Algoritmos deste tipo são úteis para resolver problemas de tamanhos relativamente pequenos.

Comportamento Assintótica

f(n)= O(n^2) (complexidade quadrática)

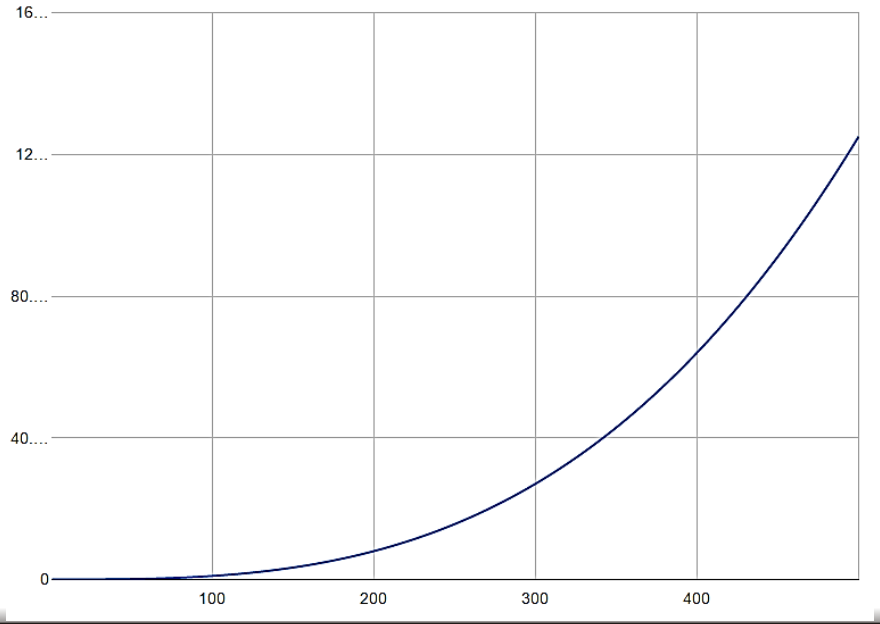


Comportamento Assintótica

f(n)= O(n^3) (complexidade cúbica)

- ✓ Algoritmos desta ordem de complexidade são úteis apenas para resolver pequenos problemas.

Exemplo: Quando n é cem, o número de operações é da ordem de 1 milhão.

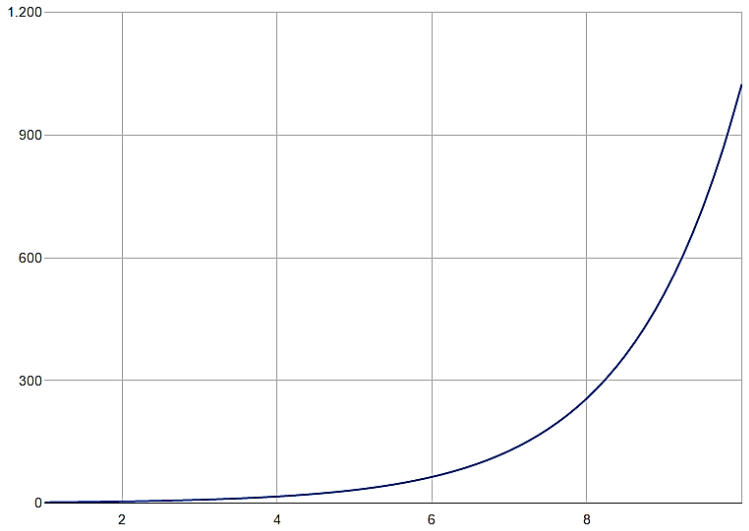


Comportamento Assintótica

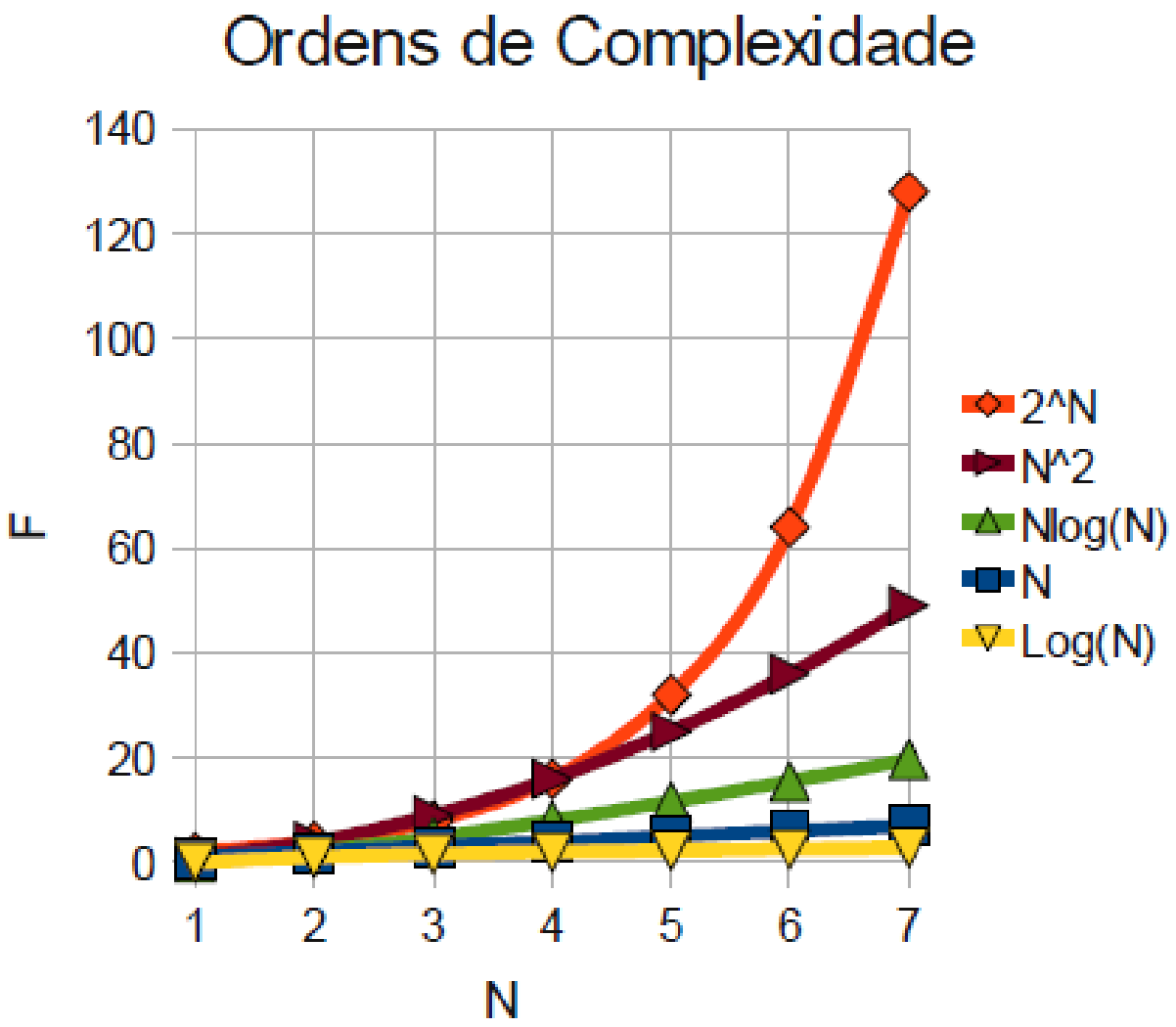
f(n)= O(2^n) (complexidade exponencial)

- ✓ Algoritmos desta ordem de complexidade geralmente não são úteis sob o ponto de vista prático. Eles ocorrem na solução de problemas quando se usa força bruta para resolvê-los.

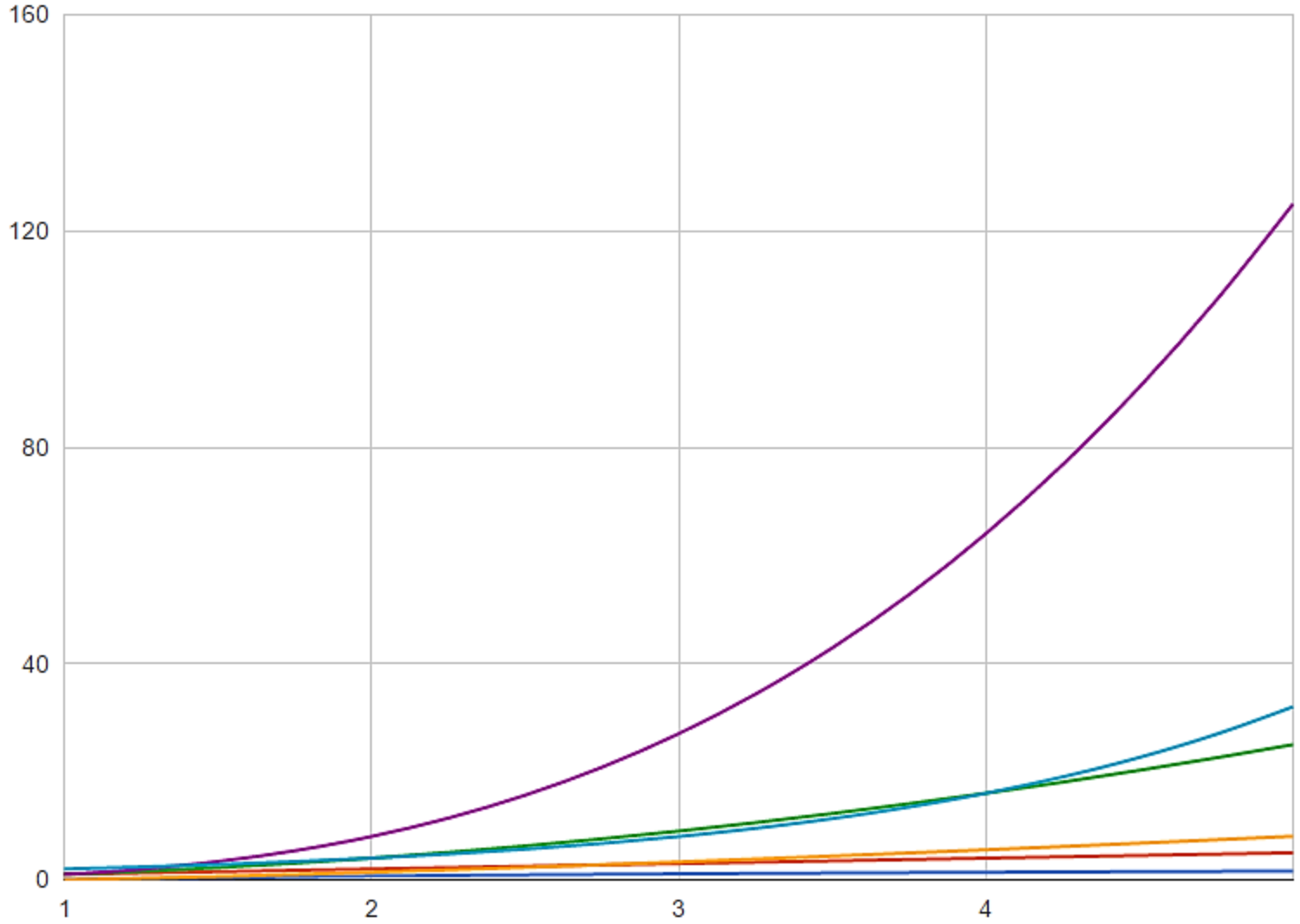
Exemplos: Quando n é vinte, o tempo de execução é cerca de um milhão.



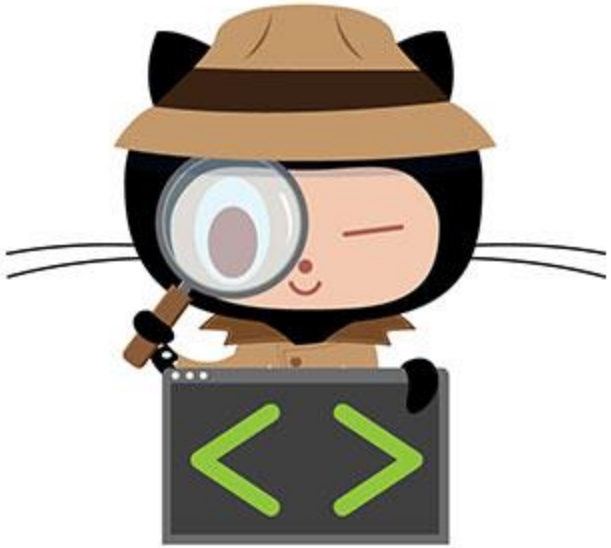
Comportamento Assintótica



Ordens de Complexidade – N = 5 para N = ∞



See you



Perguntas?