

Unidade III:

Fundamentos de Análise de Algoritmos



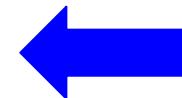
Instituto de Ciências Exatas e Informática
Departamento de Ciência da Computação

Agenda

- Potência, Logaritmo, Piso e Teto, e Função
- Contagem de operações
- Aspectos da análise de algoritmos
- Função de complexidade
- Notações O , Ω e Θ

Agenda

- **Potência, Logaritmo, Piso e Teto, e Função**
- Contagem de operações
- Aspectos da análise de algoritmos
- Função de complexidade
- Notações O , Ω e Θ



Exercício Resolvido (1)

- Resolva as equações abaixo:

a) $2^{10} =$

b) $\lg(1024) =$

c) $\lg(17) =$

d) $\lceil \lg(17) \rceil =$

e) $\lfloor \lg(17) \rfloor =$

Nota: $\lg(n)$ é a mesma coisa que o logaritmo de n na base dois, ou seja, $\log_2(n)$

Exercício Resolvido (1)

- Resolva as equações abaixo:

a) $2^{10} = \textcolor{blue}{1024}$



b) $\lg(1024) = \textcolor{blue}{10}$

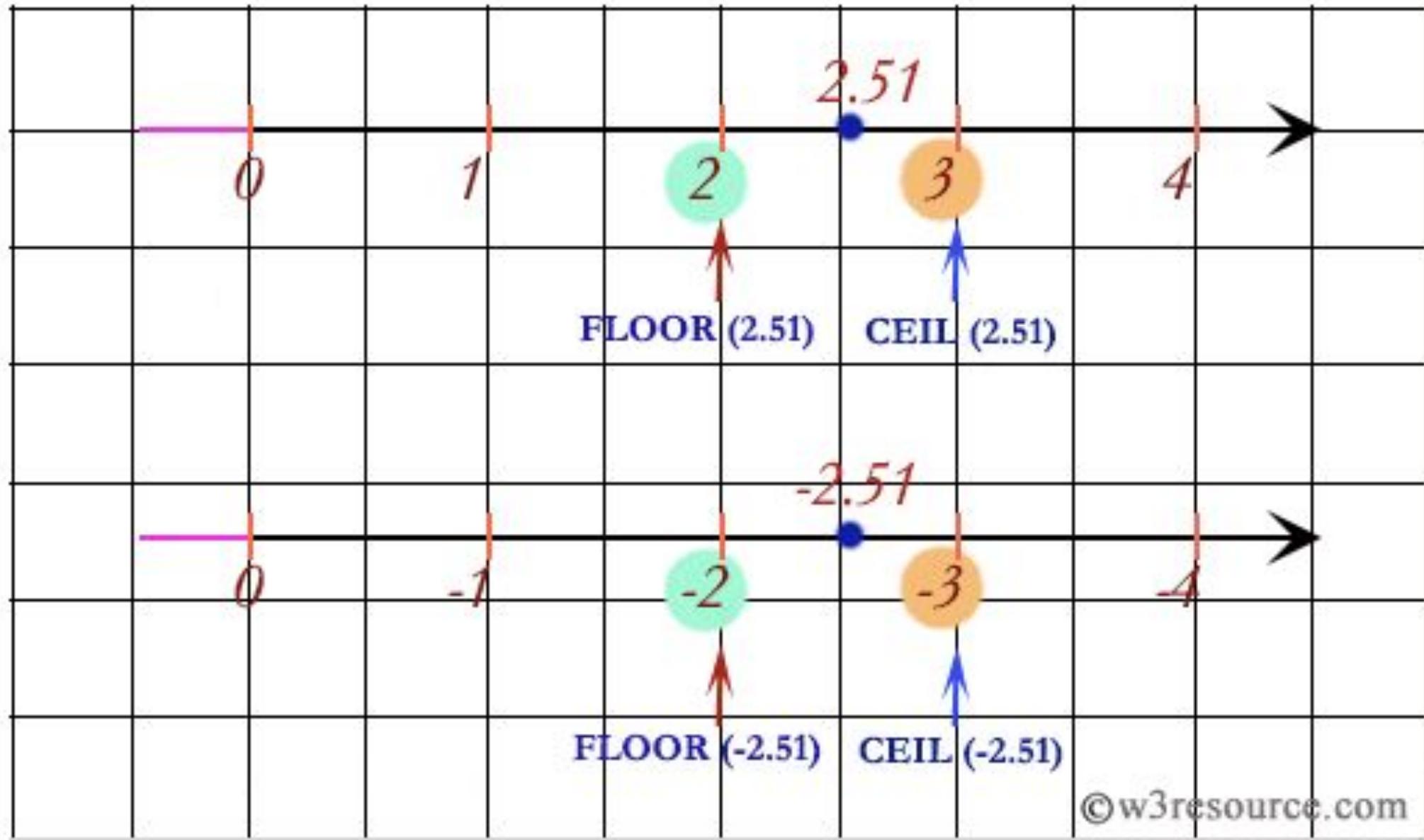
c) $\lg(17) = \textcolor{blue}{4,08746284125034}$

d) $\lceil \lg(17) \rceil = \textcolor{blue}{5}$

e) $\lfloor \lg(17) \rfloor = \textcolor{blue}{4}$

Nota: $\lg(n)$ é a mesma coisa que o logaritmo de n na base dois, ou seja, $\log_2(n)$

Piso e Teto



Exercício Resolvido (2)

• Plote um gráfico com todas as funções abaixo:

a) $f(n) = n^3$

b) $f(n) = n^2$

c) $f(n) = n \times \lg(n)$

d) $f(n) = n$

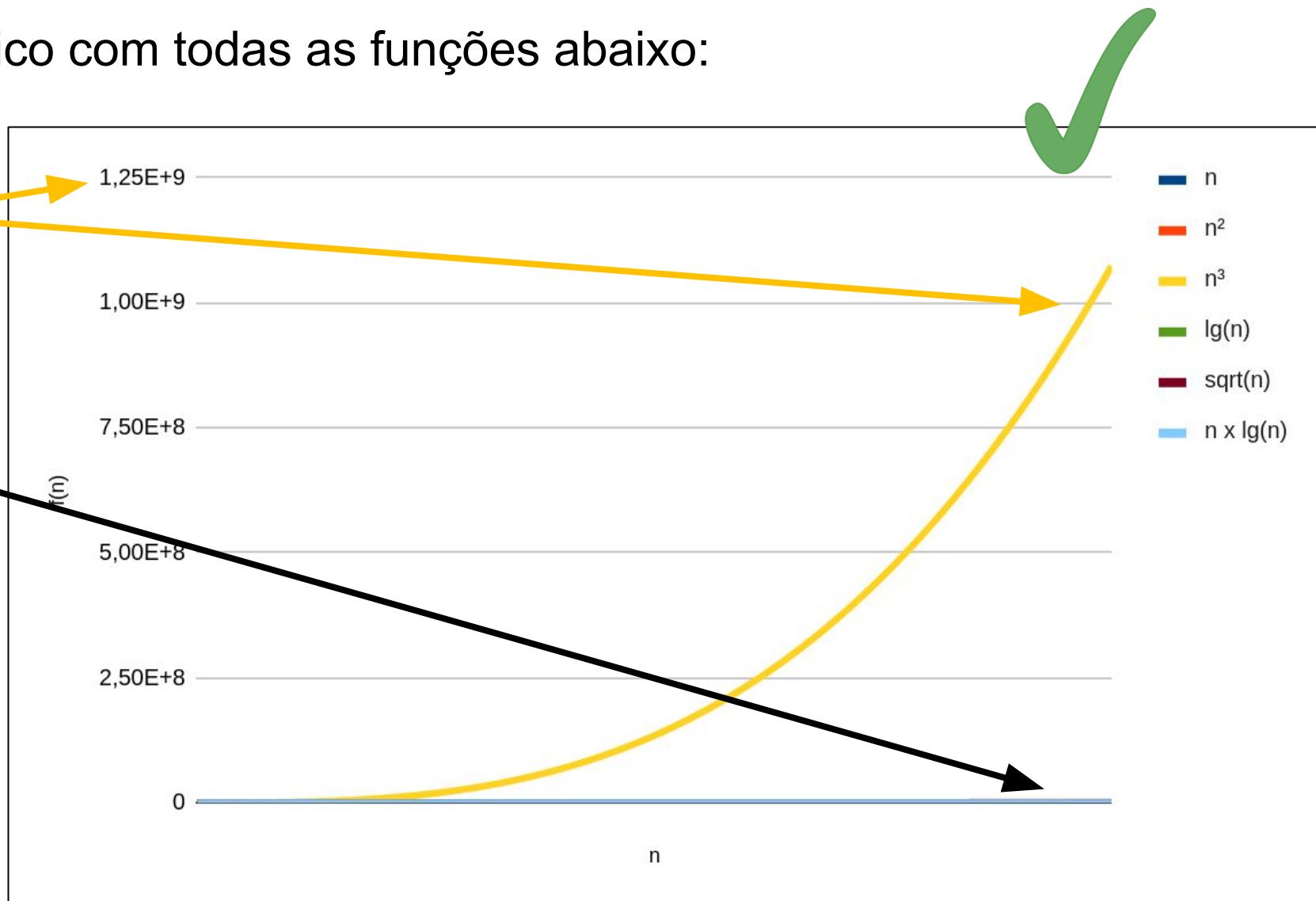
e) $f(n) = \sqrt{n}$

f) $f(n) = \lg(n)$

Exercício Resolvido (2)

- Pinte um gráfico com todas as funções abaixo:

- a) $f(n) = n^3$
- b) $f(n) = n^2$
- c) $f(n) = n \times \lg(n)$
- d) $f(n) = n$
- e) $f(n) = \sqrt{n}$
- f) $f(n) = \lg(n)$



Exercício Resolvido (2)

- Pinte um gráfico com todas as funções abaixo:

a) $f(n) = n^3$

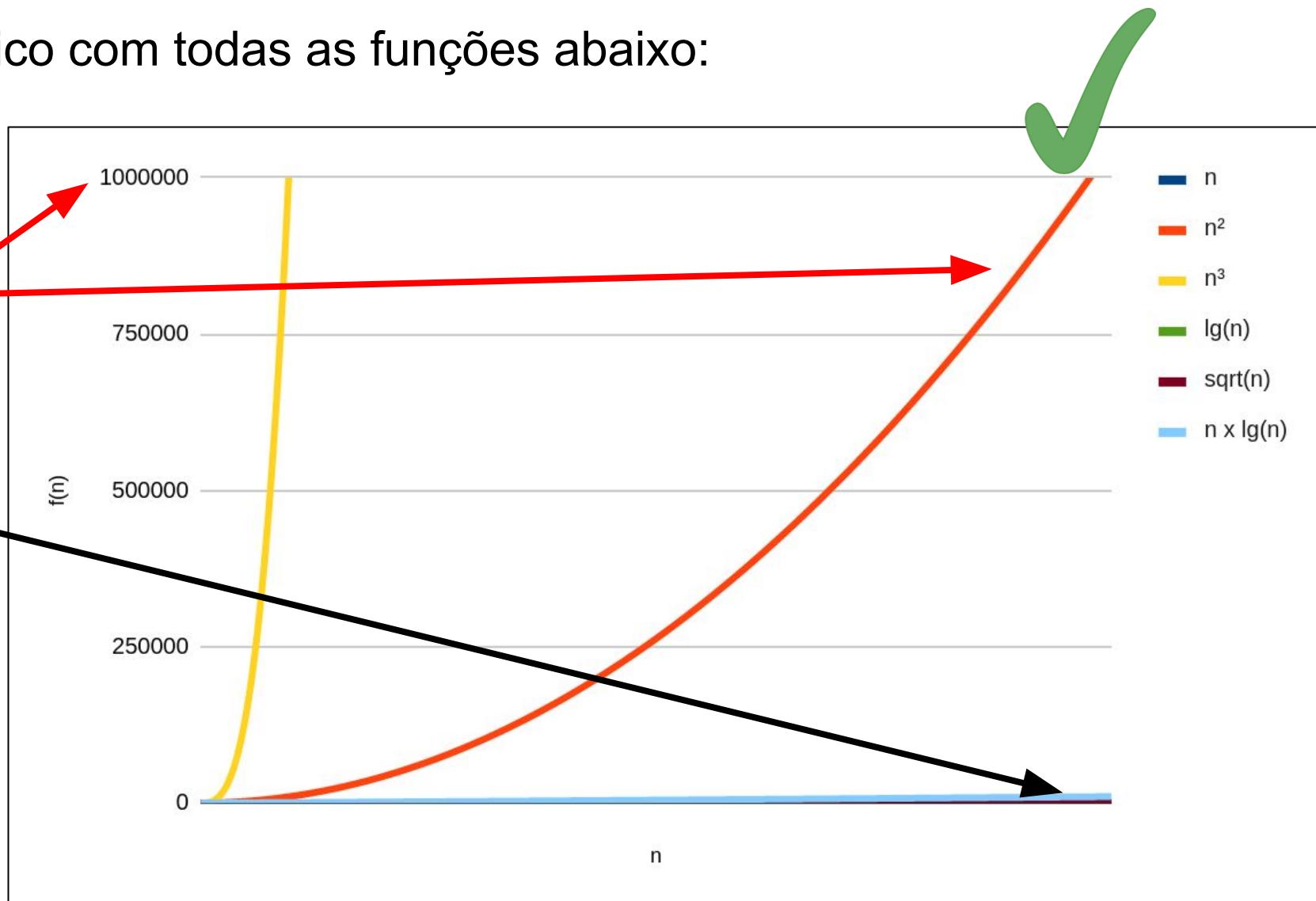
b) $f(n) = n^2$

c) $f(n) = n \times \lg(n)$

d) $f(n) = n$

e) $f(n) = \sqrt{n}$

f) $f(n) = \lg(n)$



Exercício Resolvido (2)

- Pinte um gráfico com todas as funções abaixo:

a) $f(n) = n^3$

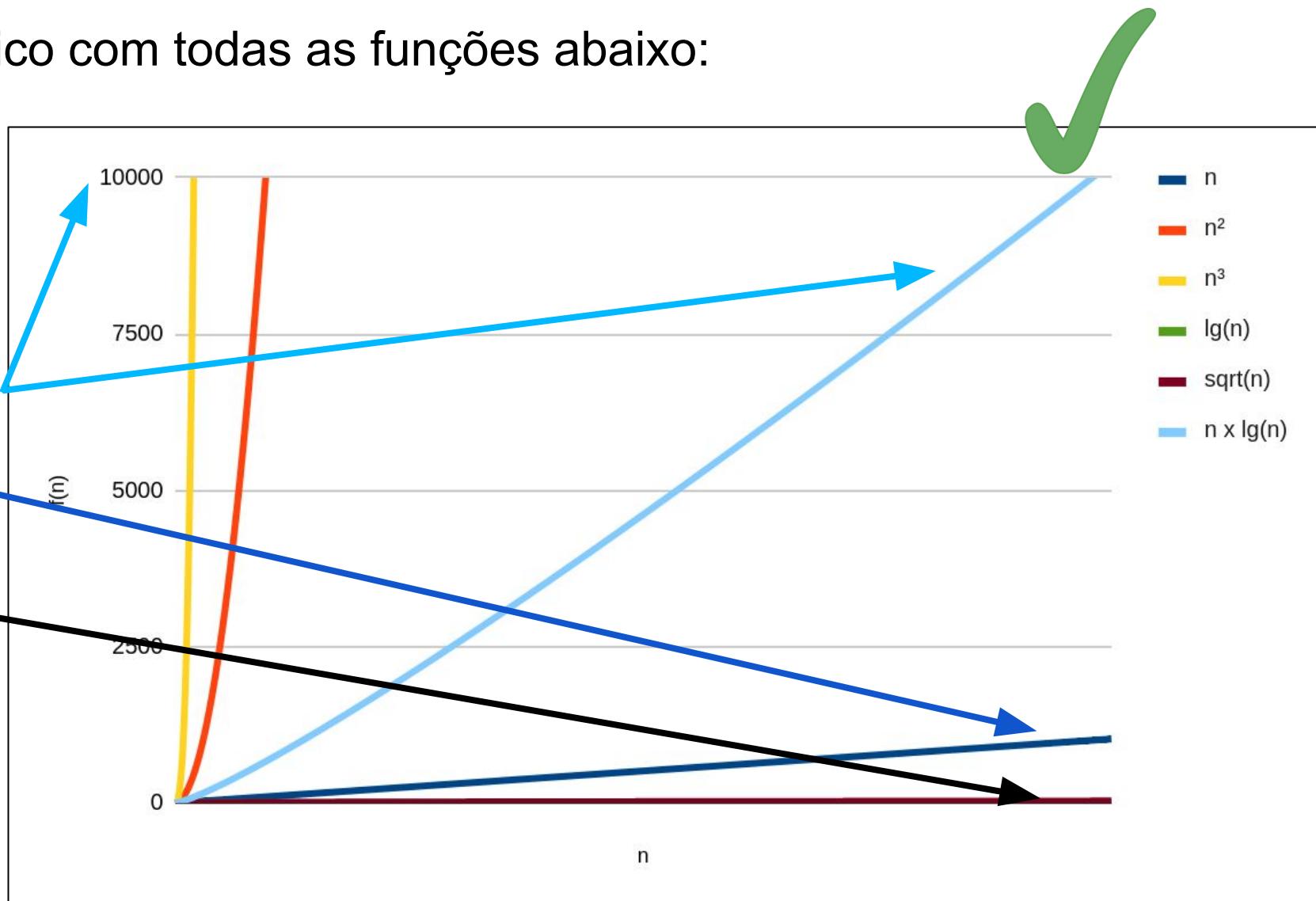
b) $f(n) = n^2$

c) $f(n) = n \times \lg(n)$

d) $f(n) = n$

e) $f(n) = \sqrt{n}$

f) $f(n) = \lg(n)$



Exercício Resolvido (2)

- Pinte um gráfico com todas as funções abaixo:

a) $f(n) = n^3$

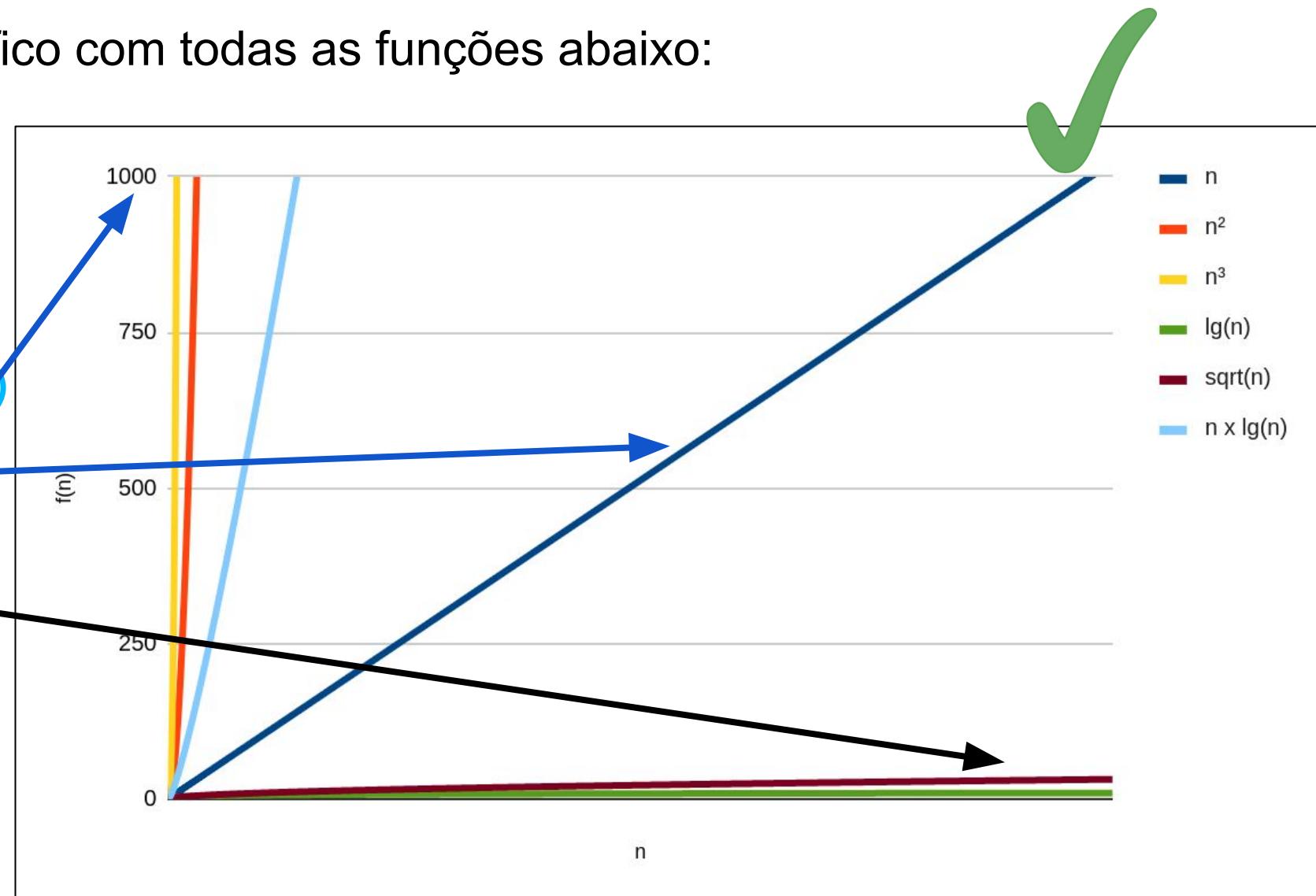
b) $f(n) = n^2$

c) $f(n) = n \times \lg(n)$

d) $f(n) = n$

e) $f(n) = \sqrt{n}$

f) $f(n) = \lg(n)$



Exercício Resolvido (2)

- Pinte um gráfico com todas as funções abaixo:

a) $f(n) = n^3$

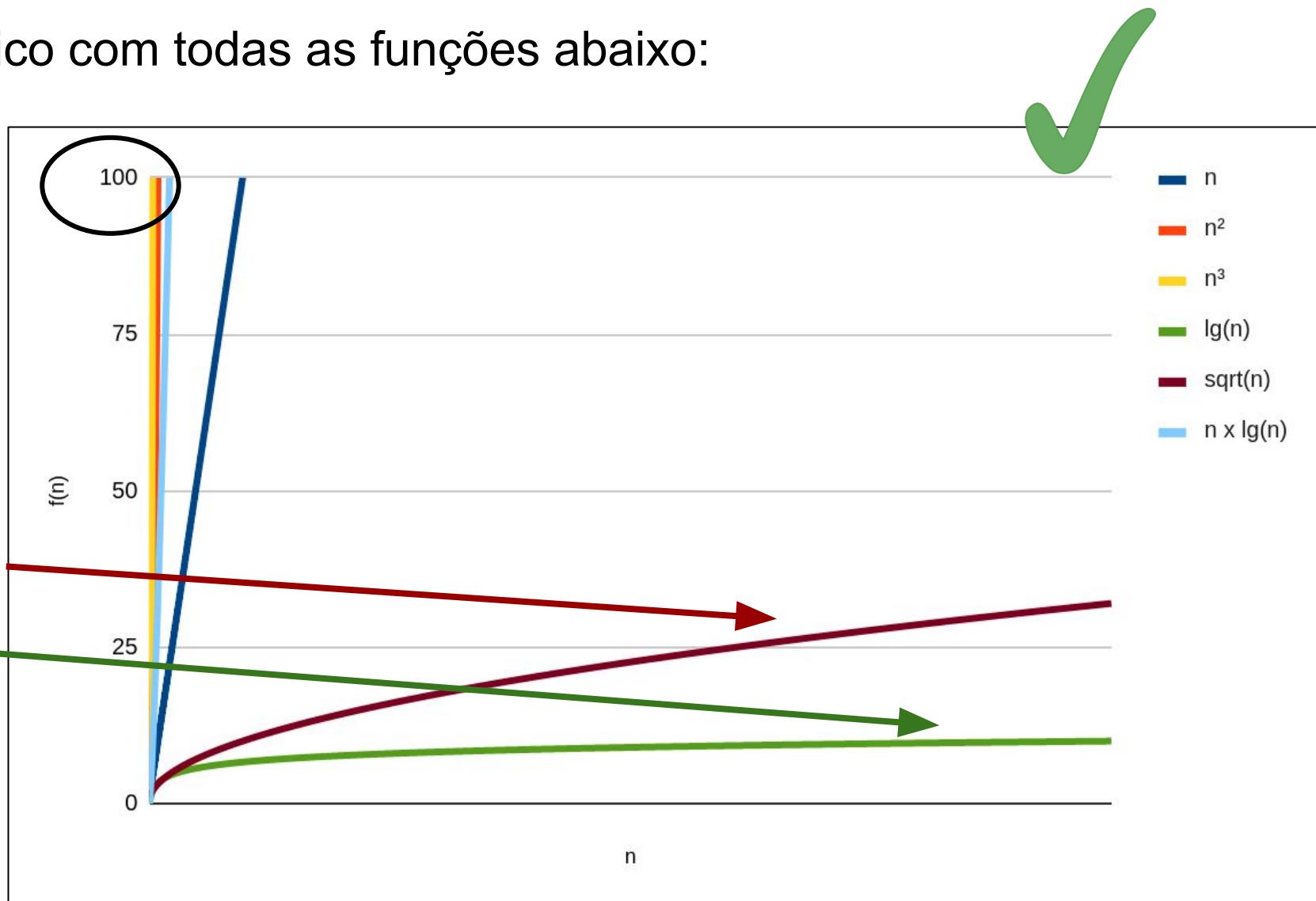
b) $f(n) = n^2$

c) $f(n) = n \times \lg(n)$

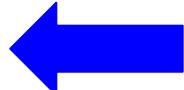
d) $f(n) = n$

e) $f(n) = \sqrt{n}$

f) $f(n) = \lg(n)$



Agenda

- Potência, Logaritmo, Piso e Teto, e Função
- **Contagem de operações** 
- Aspectos da análise de algoritmos
- Função de complexidade
- Notações O , Ω e Θ

Exercício Resolvido (3)

- Calcule o número de subtrações que o código abaixo realiza:

```
...
for (int i = 0; i < n; i++){
    if (rand() % 2 == 0){
        a--;
        b--;
    } else {
        c--;
    }
}
```

Exercício Resolvido (3)

- Calcule o número de subtrações que o código abaixo realiza:



```
...
for (int i = 0; i < n; i++){
    if (rand() % 2 == 0){
        a--;
        b--;
    } else {
        c--;
    }
}
```

//Melhor caso: $f(n) = n$, logo, $O(n)$, $\Omega(n)$ e $\Theta(n)$
//Pior caso: $f(n) = 2n$, logo, $O(n)$, $\Omega(n)$ e $\Theta(n)$

Cenários Possíveis

- **Melhor caso:** menor “tempo de execução” para todas entradas possíveis de tamanho n
- **Pior caso:** maior “tempo de execução” para todas entradas possíveis
- **Caso médio (ou esperado):** média dos tempos de execução para todas as entradas possíveis (abordado em PAA)

Contagem de Operações com Condicional

- Será o custo da condição mais ou o da lista de verdadeira ou o da falsa

```
if ( condição() ){
    listaVerdadeiro();
} else {
    listaFalso();
}
```

Melhor caso: condição() + mínimo(listaVerdadeiro(), listaFalso())

Pior caso: condição() + máximo(listaVerdadeiro(), listaFalso())

Exercício Resolvido (4)

- Calcule o número de subtrações que o código abaixo realiza:

```
...
for (int i = 3; i < n; i++){
    a--;
}
```

Exercício Resolvido (4)

- Calcule o número de subtrações que o código abaixo realiza:



```
...
for (int i = 3; i < n; i++){
    a--;
}
//n - 3 subtrações
// Logo, O(n), Ω(n) e Θ(n)
```

Se $n = 6$, temos subtrações quando i vale 3, 4, 5 ($6 - 3 = 3$, vezes)

$n = 7$

3, 4, 5, 6 ($7 - 3 = 4$ vezes)

....

$n = 10$

3, 4, 5, 6, 7, 8, 9 ($10 - 3 = 7$ vezes)

Contagem de Operações com Repetição

- Será o custo da condição mais o número de iterações multiplicado pela soma dos custos da condição e da lista a ser repetida

```
while ( condição() ){  
    lista();  
}
```

Custo: condição() + n * (lista() + condição()), onde n é o número de vezes que o laço será repetido

Contagem de Operações com Repetição

- Será o número n de iterações multiplicado pela soma dos custos da lista de comandos e da condição

```
do {  
    lista();  
} while ( condição );
```

Custo: $n \times (\text{condição} + \text{lista}())$, onde n é o número de vezes que o laço será repetido

Exercício Resolvido (5)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n; i > 0; i /= 2)  
    a *= 2;
```

Exercício Resolvido (5)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n; i > 0; i /= 2)  
    a *= 2;
```

Quando n é uma potência de 2, realizamos $\lg(n) + 1$ multiplicações



Se $n = 8$, efetuamos a multiplicação quando i vale 8, 4, 2, 1

$n = 16$, 16, 8, 4, 2, 1

$n = 32$, 32, 16, 8, 4, 2, 1

Exercício Resolvido (5)

- Calcule o número de multiplicações que o código abaixo realiza:

```
for (int i = n; i > 0; i /= 2)  
    a *= 2;
```

Para um valor qualquer de n , temos $\lfloor \lg(n) \rfloor + 1$ multiplicações, logo, $O(\lg n)$, $\Omega(\lg n)$ e $\Theta(\lg n)$

$n = 7, \dots \quad 7, 3, 1$

Se $n = 8$, efetuamos a multiplicação quando i vale $8, 4, 2, 1$

$n = 9, \dots \quad 9, 4, 2, 1$

$n = 15, \dots \quad 15, 7, 3, 1$

$n = 16, \dots \quad 16, 8, 4, 2, 1$

$n = 17, \dots \quad 17, 8, 4, 2, 1$

$n = 31, \dots \quad 31, 15, 7, 3, 1$

$n = 32, \dots \quad 32, 16, 8, 4, 2, 1$

$n = 33, \dots \quad 33, 16, 8, 4, 2, 1$

Exercício Resolvido (6)

- Outra forma de compreender o código anterior é executando o mesmo

```
class Log {  
    public static void main (String[] args) {  
        int[] n = {4,5,6,7,8,9,10,11,12,13,14,15,16,17,31,32,33,63,64,65};  
        int cont;  
  
        for(int k = 0; k < n.length; k++){  
            System.out.print("\n[n = " + n[k] + "] => ");  
            cont = 0;  
            for(int i = n[k]; i > 0; i /= 2){  
                System.out.print(" " + i);  
                cont++;  
            }  
            System.out.print(" (" + cont + " vezes)");  
        }  
        System.out.print("\n");  
    }  
}
```

Contagem de Operações com Repetição

- Quando tivermos uma estrutura de repetição em que o escopo de busca é sistematicamente dividido pela metade, temos um custo logarítmico

```
for (int i = n; i > 0; i /= 2){  
    lista();  
}
```

Exercício Resolvido (7)

- Encontre o menor valor em um array de inteiros



```
int min = array[0];  
  
for (int i = 1; i < n; i++){  
    if (min > array[i]){  
        min = array[i];  
    }  
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do array

2º) Quantas vezes ela será executada?

R: Se tivermos n elementos: $T(n) = n - 1$

Exercício Resolvido (7)

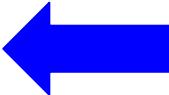
- Encontre o menor valor em um array de inteiros



```
int min = array[0];  
  
for (int i = 1; i < n; i++){  
    if (min > array[i]){  
        min = array[i];  
    }  
}
```

3º) O nosso $T(n) = n - 1$ é para qual dos três casos?

Agenda

- Potência, Logaritmo, Piso e Teto, e Função
- Contagem de operações
- **Aspectos da análise de algoritmos** 
- Função de complexidade
- Notações O , Ω e Θ

Restrição dos Algoritmos

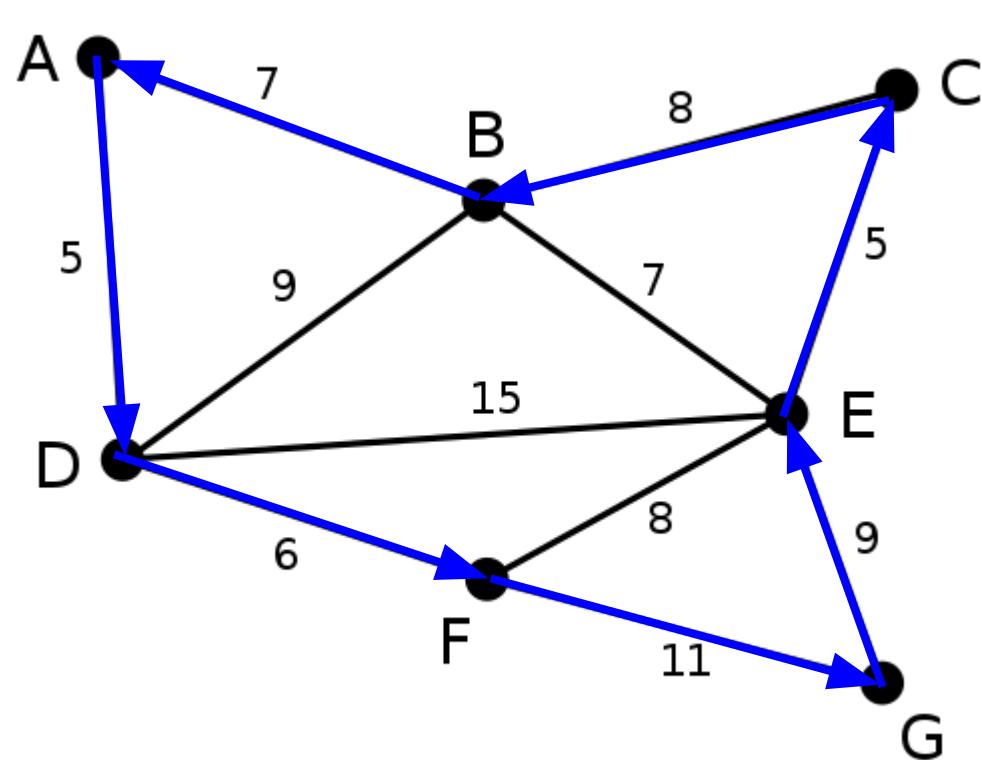
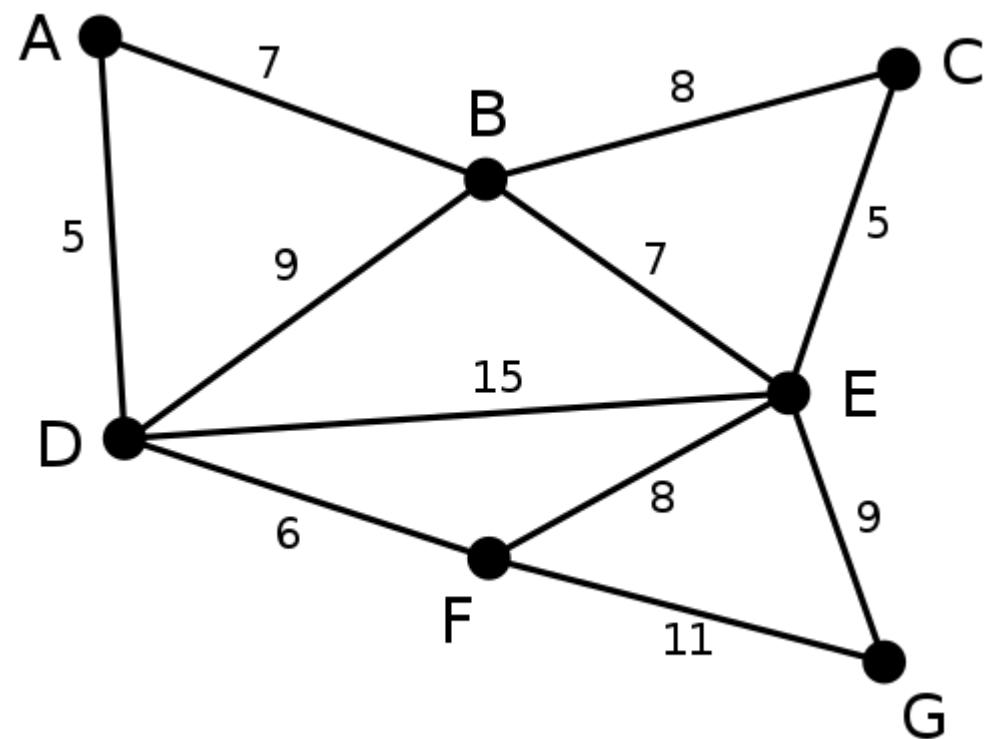
- Nossos algoritmos devem ser implementados em um computador
- Restrições do computador: capacidade computacional e armazenamento
- Logo, devemos analisar a complexidade de se implementar algoritmos

Um algoritmo que leva séculos para terminar é uma opção inadequada



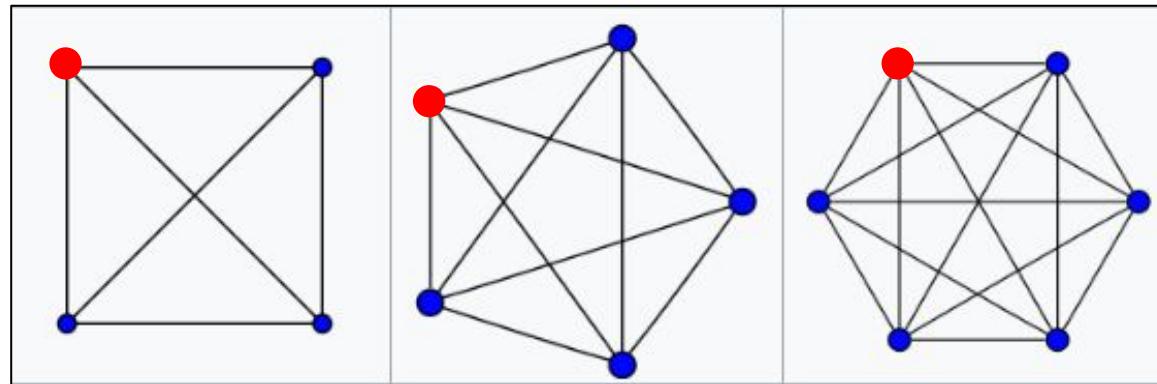
Exemplo (Rascunho) de Algoritmo NP-Completo

- Problema do Caixeiro Viajante



Exemplo (Rascunho) de Algoritmo NP-Completo

- Problema do Caixeiro Viajante



Número de combinações:

— 3 — 2 — 1

— 4 — 3 — 2 — 1

— 5 — 4 — 3 — 2 — 1

Exemplo (Rascunho) de Algoritmo NP-Completo

- Rascunho do algoritmo força bruta para encontrar a solução ótima do PCV

Número de cidades	Tempo de execução
5	5 s
6	$5 \times (5\text{s}) = 25\text{ s}$
7	$6 \times (25\text{s}) = 150\text{ s} = 2,5\text{ min}$
8	$7 \times (2,5\text{ min}) = 17,5\text{ min}$
9	$8 \times (17,5\text{ min}) = 140\text{ min} = 2,34\text{ h}$
10	$9 \times (2,34\text{ h}) = 21\text{ h}$
11	$10 \times (21\text{ dias}) = 210 = 8,75\text{ dias}$
12	$11 \times (8,75\text{ dias}) = 96,25\text{ dias}$
13	$12 \times (96,25\text{ dias}) = 1155 = 3,15\text{ anos}$
14	$13 \times (3,15\text{ anos}) = 41,02\text{ anos}$
15	$14 \times (41,02\text{ anos}) = 5,74\text{ séculos}$
16	$15 \times (5,74\text{ séculos}) = 8,6\text{ milênios}$

Exemplo (Rascunho) de Algoritmo NP-Completo

- Rascunho do algoritmo força bruta para encontrar a solução ótima do PCV

Número de cidades	Tempo de execução
-------------------	-------------------

Observação (1): Na verdade, a solução ótima para o PCV é duas vezes mais rápida que a apresentada, contudo, isso é “indiferente” na tendência de crescimento

9	$8 \times (17,5 \text{ min}) = 140 \text{ min} = 2,34 \text{ h}$
10	$9 \times (2,34 \text{ h}) = 21 \text{ h}$

Observação (2): Se tivermos um computador 100 vezes mais rápido, isso também será “indiferente” na tendência de crescimento

15	$14 \times (41,02 \text{ anos}) = 5,74 \text{ séculos}$
16	$15 \times (5,74 \text{ séculos}) = 8,6 \text{ milênios}$

Métricas para a Análise de Complexidade

- Tempo de execução
- Espaço de memória ocupado
- Outros...

Tipos de Análise de Complexidade

- **Análise de um algoritmo particular**: analisamos o custo de um algoritmo específico para um problema específico
- **Análise de uma classe de algoritmos**: analisamos o menor custo possível para resolver um problema específico
 - **Limite da família de algoritmos**, nível mínimo de dificuldade para ser resolvido

Como Medir o Custo de um Algoritmo

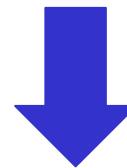


Restrições no Modelo do Cronômetro

- Hardware
- Arquitetura
- Sistema Operacional
- Linguagem
- Compilador

Exemplo de Otimização do Compilador

```
for (int i = 0; i < 20 ; i++){  
    array[i] = i;  
}
```



Qual é a vantagem de
cada um dos códigos?

```
array [0] = 0;  
array [1] = 1;
```

■ ■ ■

```
array [19] = 19;
```

Ainda sobre Otimização de Compiladores

- Frequentemente, alunos fazem otimizações desnecessárias em termos de eficiência
- Por exemplo, frequentemente, o compilador gera o mesmo código objeto para if-else-if e switch-case; for e while; entre outros...

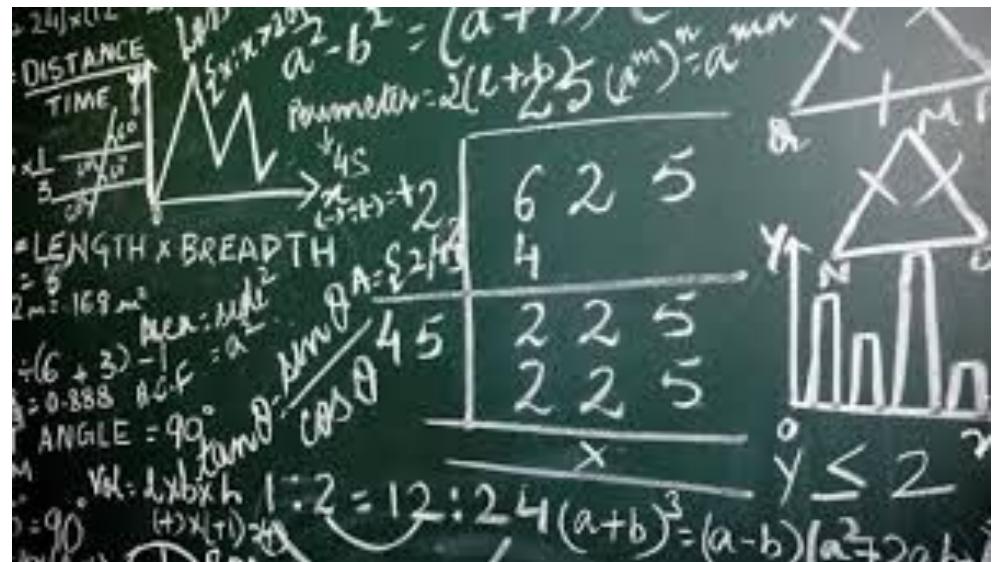
Como Medir o Custo de um Algoritmo



Como Medir o Custo de um Algoritmo

Modelo

Matemático



Modelo Matemático para Contar Operações

- Determinamos e contamos as operações relevantes. Em AEDs II, quase sempre, comparações entre registros (elementos do array)
- O custo total de um algoritmo é igual a soma do custo de suas operações
- Desconsideramos sobrecargas de gerenciamento de memória ou E/S
- A menos que dito o contrário, consideramos o pior caso
- Precisamos definir a função de complexidade

Agenda

- Potência, Logaritmo, Piso e Teto, e Função
- Contagem de operações
- Aspectos da análise de algoritmos
- **Função de complexidade** 
- Notações O , Ω e Θ

Algumas Funções de Complexidade

- **Função de complexidade de tempo** mede o tempo (número de execuções da operação relevante) de execução do algoritmo para um problema de tamanho n
- **Função de complexidade de espaço** mede a quantidade de memória necessária para executar um algoritmo de tamanho n

Como Calcular a Complexidade de um Algoritmo



Como Calcular a Complexidade de um Algoritmo

- Da mesma forma que calculamos o custo de um churrasco:
 - Carne: 400 gramas por pessoa (preço médio do kg R\$ 20,00 - picanha, asinha, coraçãozinho ...)
 - Cerveja: 1,2 litros por pessoa (litro R\$ 3,80)
 - Refrigerante: 1 litro por pessoa (Garrafa 2 litros R\$ 3,50)

Exercício: Monte a função de complexidade (ou custo) do nosso churrasco.

Como Calcular a Complexidade de um Algoritmo

- Da mesma forma que calculamos o custo de um churrasco:
 - Carne: 400 gramas por pessoa (preço médio do kg R\$ 20,00 - picanha, asinha, coraçãozinho ...)
 - Cerveja: 1,2 litros por pessoa (litro R\$ 3,80)
 - Refrigerante: 1 litro por pessoa (Garrafa 2 litros R\$ 3,50)

Exercício: Monte a função de complexidade (ou custo) do nosso churrasco.

$$\begin{aligned}f(n) &= n * \frac{400}{1000} * 20 + n * 1,2 * 3,8 + n * 1 * \frac{3,5}{2} \\&= 14,31 * n\end{aligned}$$

Como Calcular a Complexidade de um Algoritmo

- Da mesma forma que calculamos o custo de uma viagem:
 - Passagem:
 - Hotel:
 - Saídas:

Cálculo de Complexidade para Condicional

- Será o custo da condição mais ou o da lista de verdadeira ou o da falsa

```
if ( condição() ){  
    listaVerdadeiro();  
} else {  
    listaFalso();  
}
```

Melhor caso: condição() + mínimo(listaVerdadeiro(), listaFalso())

Pior caso: condição() + máximo(listaVerdadeiro(), listaFalso())

Cálculo de Complexidade para Repetição

- Será o custo da condição mais o número de interações multiplicado pela soma dos custos da condição e da lista a ser repetida

```
while ( condição() ){  
    lista();  
}
```

Custo: condição() + n * (lista() + condição()), onde n é o número de vezes que o laço será repetido

Cálculo de Complexidade

- Outros laços: sempre consideramos o limite superior
- Métodos: consideramos o custo do método
- Métodos recursivos: utilizamos equações de recorrência (Teoria dos Grafos e Computabilidade)

Algoritmo Ótimo

- Algoritmo cujo custo é igual ao menor custo possível

Exercício Resolvido (8): Encontrar Mínimo

```
int min = array[0];  
  
for (int i = 1; i < n; i++){  
    if (min > array[i]){  
        min = array[i];  
    }  
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do *array*

2º) Quantas vezes ela será executada?

R: Se tivermos n elementos: $T(n) = n - 1$

3º) O nosso $T(n) = n - 1$ é para qual dos três casos?

R: Para os três casos

4º) O nosso algoritmo é ótimo? Por que?

Exercício Resolvido (8): Encontrar Mínimo

```
int min = array[0];  
  
for (int i = 1; i < n; i++){  
    if (min > array[i]){  
        min = array[i];  
    }  
}
```

1º) Qual é a operação relevante?



R: Comparação entre elementos do array

2º) Quantas vezes ela será executada?

R: Se tivermos n elementos: $T(n) = n - 1$

3º) O nosso $T(n) = n - 1$ é para qual dos três casos?

R: Para os três casos

4º) O nosso algoritmo é ótimo? Por que?

R: Sim porque temos que testar todos os elementos para garantir nossa resposta

Exercício Resolvido (9): Pesquisa Sequencial

```
boolean resp = false;  
  
for (int i = 0; i < n; i++){  
    if (array[i] == x){  
        resp = true;  
        i = n;  
    }  
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do array

2º) Quantas vezes ela será executada?

R: Melhor caso: $f(n) = 1$

Pior caso: $f(n) = n$

Caso médio: $f(n) = (n + 1) / 2$

3º) O nosso algoritmo é ótimo? Por que?

Exercício Resolvido (9): Pesquisa Sequencial

```
boolean resp = false;  
  
for (int i = 0; i < n; i++){  
    if (array[i] == x){  
        resp = true;  
        i = n;  
    }  
}
```

1º) Qual é a operação relevante?

R: Comparação entre elementos do array

2º) Quantas vezes ela será executada?

R: Melhor caso: $f(n) = 1$

Pior caso: $f(n) = n$

Caso médio: $f(n) = (n + 1) / 2$

3º) O nosso algoritmo é ótimo? Por que?

R: Sim porque temos que testar todos os elementos para garantir nossa resposta



Exercício (1)

- Encontre o maior e menor valores em um *array* de inteiros e, em seguida, encontre a função de complexidade de tempo para sua solução

Exercício (2)

- Considerando o problema de encontrar o maior e menor valores em um array de inteiros, veja os quatro códigos propostos e analisados no livro do Ziviani

Exercício Resolvido (10)

- Um aluno deve procurar um valor em um *array* de números reais. Ele tem duas alternativas. Primeiro, executar uma pesquisa sequencial. Segundo, ordenar o *array* e, em seguida, aplicar uma pesquisa binária. O que fazer?

Exercício Resolvido (10)

- Um aluno deve procurar um valor em um *array* de números reais. Ele tem duas alternativas. Primeiro, executar uma pesquisa sequencial. Segundo, ordenar o *array* e, em seguida, aplicar uma pesquisa binária. O que fazer?



O aluno deve escolher a primeira opção, pois a pesquisa sequencial tem custo $\Theta(n)$. A segunda opção tem custo $\Theta(n * \lg n)$ para ordenar mais $\Theta(\lg n)$ para a pesquisa binária

Agenda

- Potência, Logaritmo, Piso e Teto, e Função
- Contagem de operações
- Aspectos da análise de algoritmos
- Função de complexidade
- **Notações O , Ω e Θ** 

Noção sobre as Notações O , Ω e Θ

- Regras gerais
- Operações
- Definições

Regras Gerais das Notações O , Ω e Θ

- Consideramos apenas a maior potência
- Ignoramos os coeficientes

Diferença entre as Notações O , Ω e Θ

- O é o limite superior
- Ω é o limite inferior
- Θ é o limite justo

Diferença entre as Notações O , Ω e Θ

- **O é o limite superior**, logo, se um algoritmo é $O(f(n))$, ele também será $O(g(n))$ para toda função $g(n)$ tal que “ $g(n)$ é **maior** que $f(n)$ ”
- **Ω é o limite inferior**, logo, se um algoritmo é $\Omega(f(n))$, ele também será $\Omega(g(n))$ para toda função $g(n)$ tal que “ $g(n)$ é **menor** que $f(n)$ ”
- **Θ é o limite justo**, logo, $g(n)$ é $O(f(n))$ and $\Omega(f(n))$ se e somente se $g(n)$ é $\Theta(f(n))$

Exercício Resolvido (11)

• Responda se as afirmações são verdadeiras ou falsas:

a) $3n^2 + 5n + 1$ é $O(n)$:

b) $3n^2 + 5n + 1$ é $O(n^2)$:

c) $3n^2 + 5n + 1$ é $O(n^3)$:

d) $3n^2 + 5n + 1$ é $\Omega(n)$:

e) $3n^2 + 5n + 1$ é $\Omega(n^2)$:

f) $3n^2 + 5n + 1$ é $\Omega(n^3)$:

g) $3n^2 + 5n + 1$ é $\Theta(n)$:

h) $3n^2 + 5n + 1$ é $\Theta(n^2)$:

i) $3n^2 + 5n + 1$ é $\Theta(n^3)$:

Exercício Resolvido (11)

- Responda se as afirmações são verdadeiras ou falsas:

a) $3n^2 + 5n + 1$ é $O(n)$:

b) $3n^2 + 5n + 1$ é $O(n^2)$: **verdadeira**

c) $3n^2 + 5n + 1$ é $O(n^3)$:

d) $3n^2 + 5n + 1$ é $\Omega(n)$:

e) $3n^2 + 5n + 1$ é $\Omega(n^2)$: **verdadeira**

f) $3n^2 + 5n + 1$ é $\Omega(n^3)$:

g) $3n^2 + 5n + 1$ é $\Theta(n)$:

h) $3n^2 + 5n + 1$ é $\Theta(n^2)$: **verdadeira**

i) $3n^2 + 5n + 1$ é $\Theta(n^3)$:



Exercício Resolvido (11)

- Responda se as afirmações são verdadeiras ou falsas:

- a) $3n^2 + 5n + 1$ é $O(n)$:
- b) $3n^2 + 5n + 1$ é $O(n^2)$: verdadeira
- c) $3n^2 + 5n + 1$ é $O(n^3)$: verdadeira
- d) $3n^2 + 5n + 1$ é $\Omega(n)$: verdadeira
- e) $3n^2 + 5n + 1$ é $\Omega(n^2)$: verdadeira
- f) $3n^2 + 5n + 1$ é $\Omega(n^3)$:
- g) $3n^2 + 5n + 1$ é $\Theta(n)$:
- h) $3n^2 + 5n + 1$ é $\Theta(n^2)$: verdadeira
- i) $3n^2 + 5n + 1$ é $\Theta(n^3)$:



Exercício Resolvido (11)

- Responda se as afirmações são verdadeiras ou falsas:

- a) $3n^2 + 5n + 1$ é $O(n)$: **falsa**
- b) $3n^2 + 5n + 1$ é $O(n^2)$: verdadeira
- c) $3n^2 + 5n + 1$ é $O(n^3)$: verdadeira
- d) $3n^2 + 5n + 1$ é $\Omega(n)$: verdadeira
- e) $3n^2 + 5n + 1$ é $\Omega(n^2)$: verdadeira
- f) $3n^2 + 5n + 1$ é $\Omega(n^3)$: **falsa**
- g) $3n^2 + 5n + 1$ é $\Theta(n)$:
- h) $3n^2 + 5n + 1$ é $\Theta(n^2)$: verdadeira
- i) $3n^2 + 5n + 1$ é $\Theta(n^3)$:



Exercício Resolvido (11)

• Responda se as afirmações são verdadeiras ou falsas:

- a) $3n^2 + 5n + 1$ é $O(n)$: falsa
- b) $3n^2 + 5n + 1$ é $O(n^2)$: verdadeira
- c) $3n^2 + 5n + 1$ é $O(n^3)$: verdadeira
- d) $3n^2 + 5n + 1$ é $\Omega(n)$: verdadeira
- e) $3n^2 + 5n + 1$ é $\Omega(n^2)$: verdadeira
- f) $3n^2 + 5n + 1$ é $\Omega(n^3)$: falsa
- g) **$3n^2 + 5n + 1$ é $\Theta(n)$: falsa**
- h) $3n^2 + 5n + 1$ é $\Theta(n^2)$: verdadeira
- i) **$3n^2 + 5n + 1$ é $\Theta(n^3)$: falsa**



Exercício (3)

- Preencha verdadeiro ou falso na tabela abaixo:

	$O(1)$	$O(\lg n)$	$O(n)$	$O(n \cdot \lg(n))$	$O(n^2)$	$O(n^3)$	$O(n^5)$	$O(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \cdot \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

Exercício (4)

- Preencha verdadeiro ou falso na tabela abaixo:

	$\Omega(1)$	$\Omega(\lg n)$	$\Omega(n)$	$\Omega(n \cdot \lg(n))$	$\Omega(n^2)$	$\Omega(n^3)$	$\Omega(n^5)$	$\Omega(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \cdot \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

Exercício (5)

- Preencha verdadeiro ou falso na tabela abaixo:

	$\Theta(1)$	$\Theta(\lg n)$	$\Theta(n)$	$\Theta(n \cdot \lg(n))$	$\Theta(n^2)$	$\Theta(n^3)$	$\Theta(n^5)$	$\Theta(n^{20})$
$f(n) = \lg(n)$								
$f(n) = n \cdot \lg(n)$								
$f(n) = 5n + 1$								
$f(n) = 7n^5 - 3n^2$								
$f(n) = 99n^3 - 1000n^2$								
$f(n) = n^5 - 99999n^4$								

Operações com as Notações O , Ω e Θ

$$1) f(n) = O(f(n))$$

$$2) c \times O(f(n)) = O(f(n))$$

$$3) O(f(n)) + O(f(n)) = O(f(n))$$

$$4) O(O(f(n))) = O(f(n))$$

$$5) O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$$

$$6) O(f(n)) \times O(g(n)) = O(f(n) \times g(n))$$

$$7) f(n) \times O(g(n)) = O(f(n) \times g(n))$$

***) As mesmas propriedades são aplicadas para Ω e Θ**

Exercício Resolvido (12)

Sabendo que o Algoritmo de Seleção faz $\Theta(n^2)$ comparações entre registros, quantas dessas comparações temos no código abaixo? Justifique

```
for (int i = 0; i < n; i++){
    seleção();
}
```

Exercício Resolvido (12)

Sabendo que o Algoritmo de Seleção faz $\Theta(n^2)$ comparações entre registros, quantas dessas comparações temos no código abaixo? Justifique



```
for (int i = 0; i < n; i++) {  
    seleção();  
}
```

Neste caso, executamos o Seleção n vezes: $n \times \Theta(n^2) = \Theta(n^3)$

Exercício Resolvido (13)

Sabendo que o limite inferior da ordenação é $\Theta(n \cdot \lg n)$ e que o custo da pesquisa binária é $\Theta(\lg n)$, qual é a ordem de complexidade de uma solução em que ordenamos um array e efetuamos uma pesquisa binária. Justifique sua resposta

Exercício Resolvido (13)

Sabendo que o limite inferior da ordenação é $\Theta(n \cdot \lg n)$ e que o custo da pesquisa binária é $\Theta(\lg n)$, qual é a ordem de complexidade de uma solução em que ordenamos um array e efetuamos uma pesquisa binária. Justifique sua resposta



Neste caso, temos duas etapas e o custo total será a soma das mesmas, logo: $\Theta(n \cdot \lg n) + \Theta(\lg n) = \Theta(n \cdot \lg n)$

Exercício Resolvido (14)

- Dado $f(n) = 3n^2 - 5n - 9$, $g(n) = n \cdot \lg(n)$, $l(n) = n \cdot \lg^2(n)$ e $h(n) = 99n^8$, qual é a ordem de complexidade das operações abaixo.

Mostre sua resposta usando as notações O , Ω e Θ :

- a) $h(n) + g(n) - f(n)$
- b) $\Theta(h(n)) + \Theta(g(n)) - \Theta(f(n))$
- c) $f(n) \times g(n)$
- d) $g(n) \times l(n) + h(n)$
- e) $f(n) \times g(n) \times l(n)$
- f) $\Theta(\Theta(\Theta(\Theta(f(n))))))$

Exercício Resolvido (14)

- Dado $f(n) = 3n^2 - 5n - 9$, $g(n) = n \cdot \lg(n)$, $l(n) = n \cdot \lg^2(n)$ e $h(n) = 99n^8$, qual é a ordem de complexidade das operações abaixo.

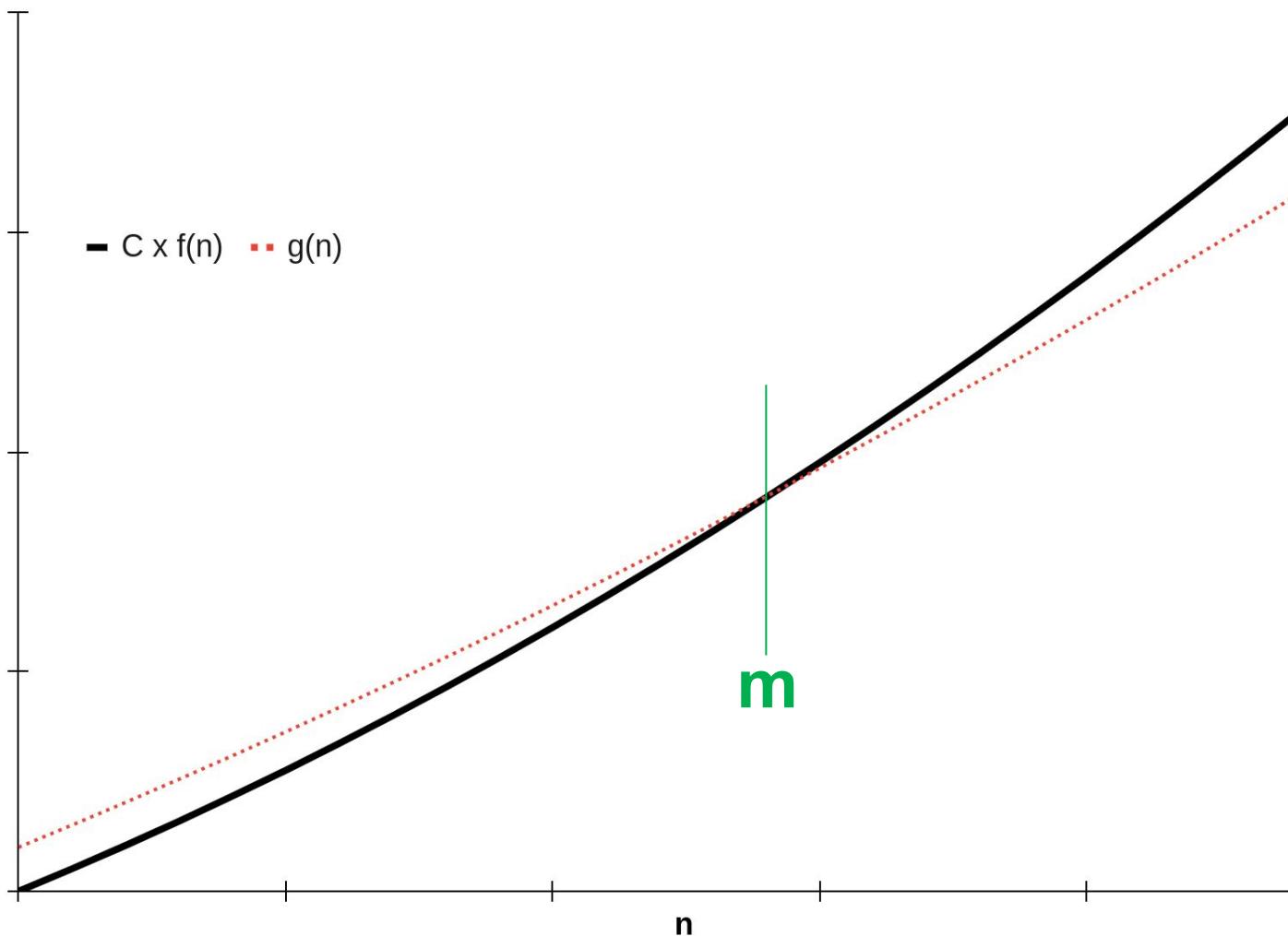
Mostre sua resposta usando as notações O , Ω e Θ :



- $h(n) + g(n) - f(n) \Rightarrow [99n^8] + [n \cdot \lg(n)] - [3n^2 - 5n - 9] \Rightarrow O(n^8), \Omega(n^8) \text{ e } \Theta(n^8)$
- $\Theta(h(n)) + \Theta(g(n)) - \Theta(f(n)) \Rightarrow \Theta(n^8) + \Theta(n \cdot \lg(n)) - \Theta(n^2) \Rightarrow O(n^8), \Omega(n^8) \text{ e } \Theta(n^8)$
- $f(n) \times g(n) \Rightarrow \Theta(n^2) \times \Theta(n \cdot \lg(n)) \Rightarrow O(n^3 \cdot \lg(n)), \Omega(n^3 \cdot \lg(n)) \text{ e } \Theta(n^3 \cdot \lg(n))$
- $g(n) \times l(n) + h(n) \Rightarrow \Theta(n \cdot \lg(n)) \times \Theta(n \cdot \lg^2(n)) + \Theta(n^8) \Rightarrow O(n^8), \Omega(n^8) \text{ e } \Theta(n^8)$
- $f(n) \times g(n) \times l(n) \Rightarrow \Theta(n^2) \times \Theta(n \cdot \lg(n)) \times \Theta(n \cdot \lg^2(n)) \Rightarrow O(n^4 \cdot \lg^3(n)), \Omega(n^4 \cdot \lg^3(n)) \text{ e } \Theta(n^4 \cdot \lg^3(n))$
- $\Theta(\Theta(\Theta(\Theta(f(n))))) \Rightarrow O(n^2), \Omega(n^2) \text{ e } \Theta(n^2)$

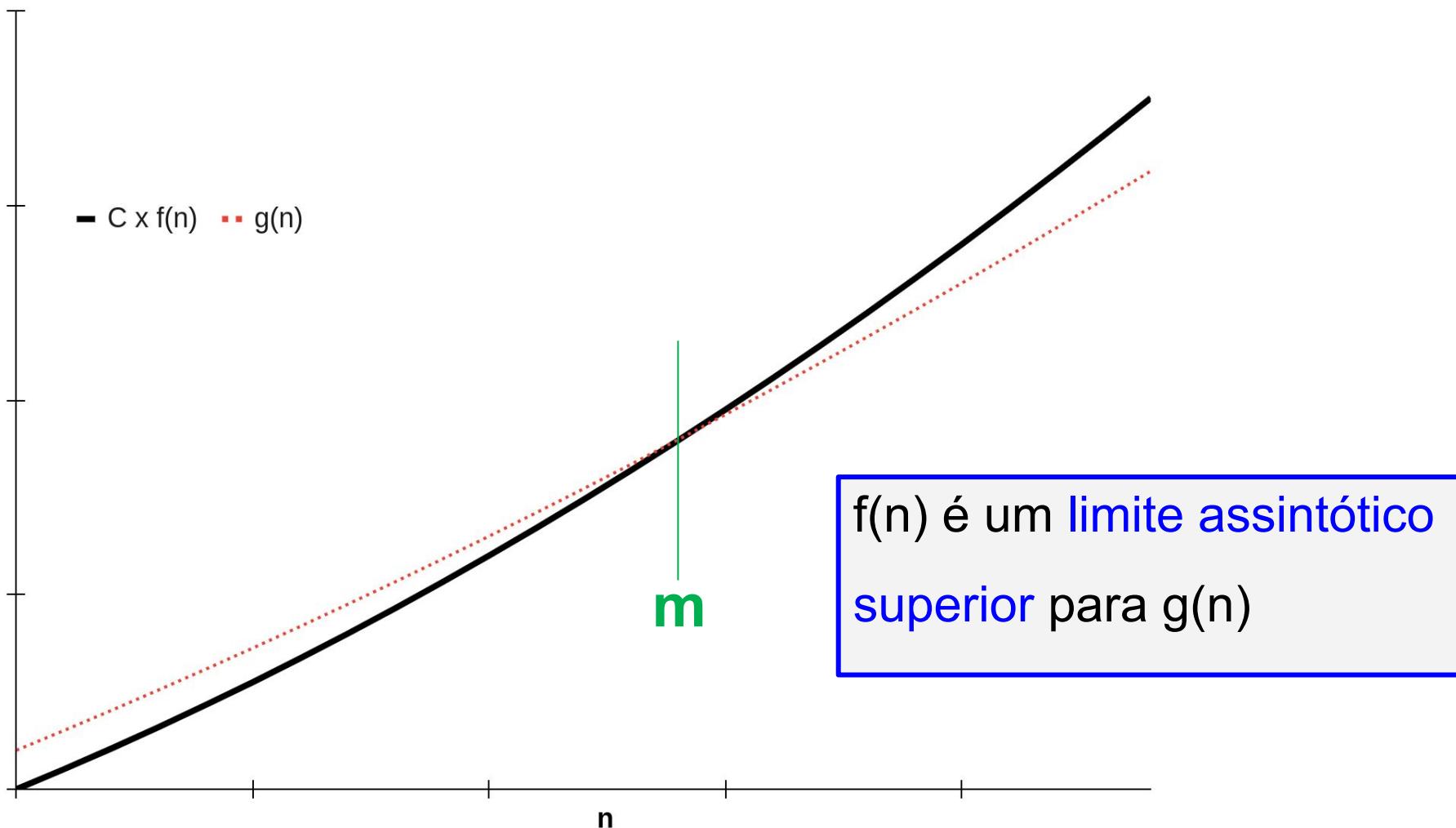
Definição da Notação O

- $g(n)$ é $O(f(n))$, se existirem as constantes positivas c e m tais que, para $n \geq m$, temos que $|g(n)| \leq c \times |f(n)|$



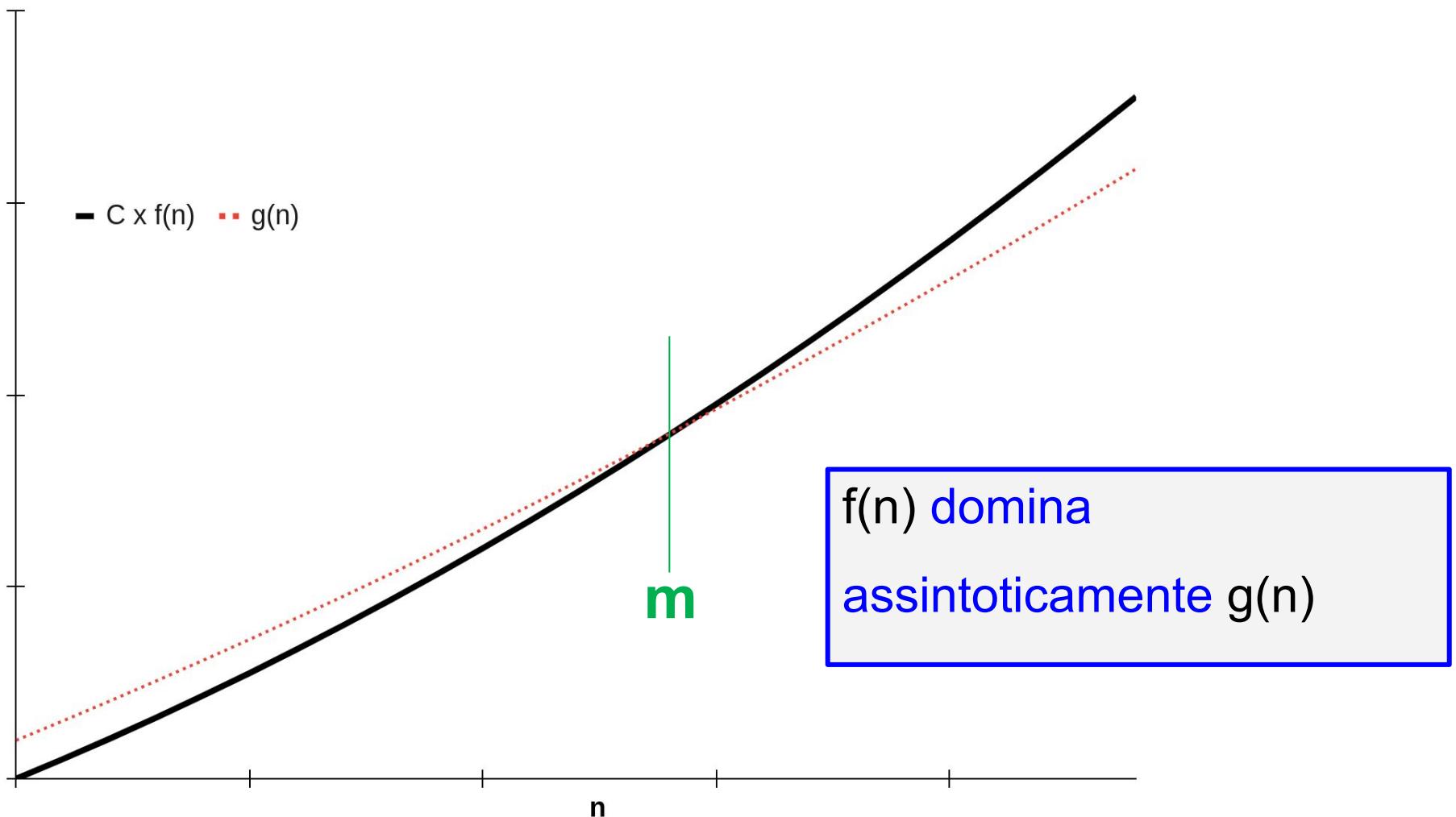
Definição da Notação O

- **g(n) é O(f(n))**, se existirem as constantes positivas c e m tais que, para $n \geq m$, temos que $|g(n)| \leq c \times |f(n)|$



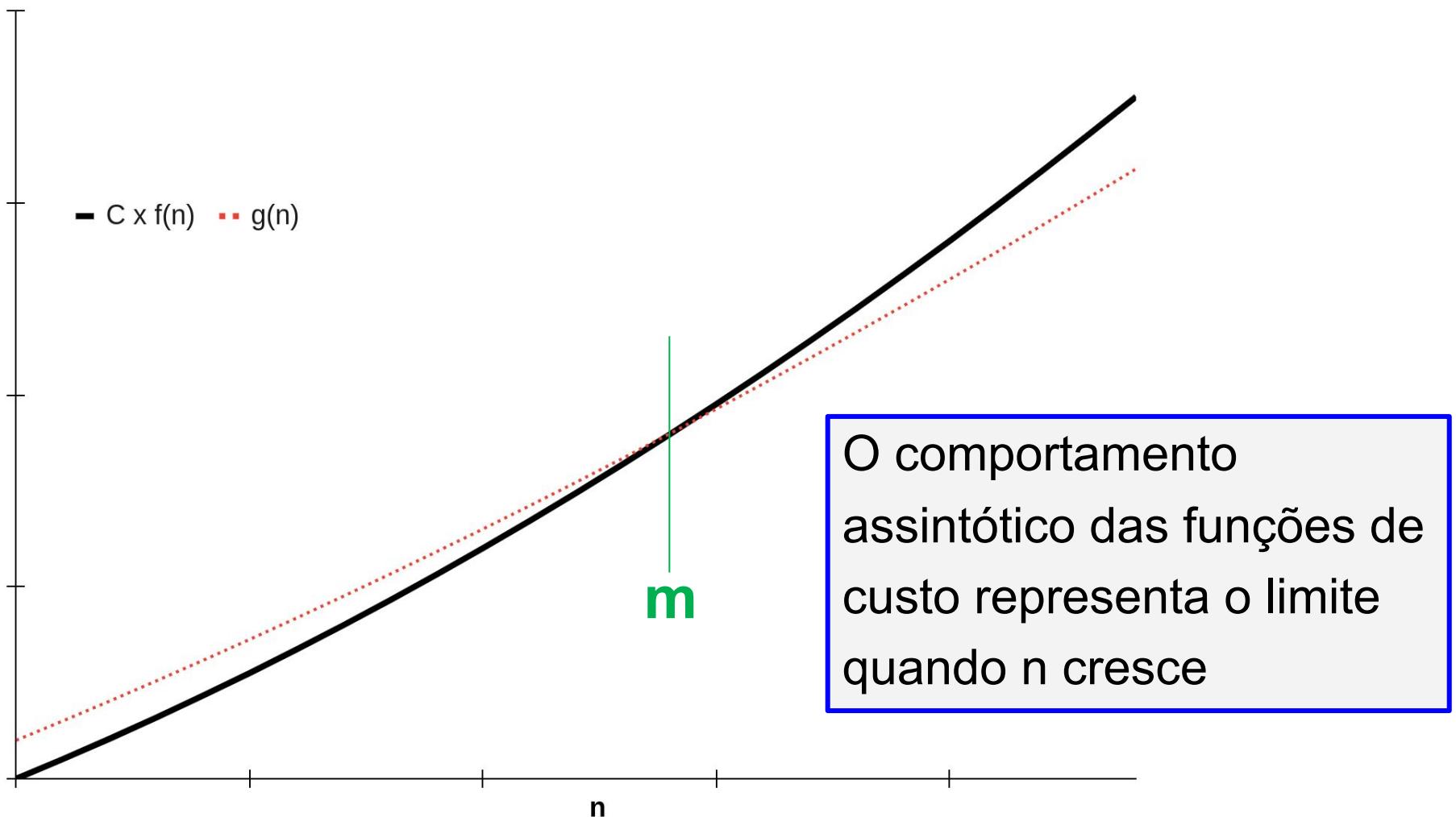
Definição da Notação O

- **g(n) é O(f(n))**, se existirem as constantes positivas c e m tais que, para $n \geq m$, temos que $|g(n)| \leq c \times |f(n)|$



Definição da Notação O

- $g(n)$ é $O(f(n))$, se existirem as constantes positivas c e m tais que, para $n \geq m$, temos que $|g(n)| \leq c \times |f(n)|$



Exercício Resolvido (15)

• Dada a definição da notação O:

- a) Mostre os valores de c e m tal que, para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n^2|$, provando que $3n^2 + 5n + 1$ é $O(n^2)$
- b) Mostre os valores de c e m tal que, para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n^3|$, provando que $3n^2 + 5n + 1$ é $O(n^3)$
- c) Prove que $3n^2 + 5n + 1$ não é $O(n)$

Exercício Resolvido (15)

• Dada a definição da notação O:

- a) Mostre os valores de c e m tal que, para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n^2|$,
provando que $3n^2 + 5n + 1$ é $O(n^2)$



Para que tal inequação seja verdadeira, c tem que ser maior do que três (e.g., quatro)

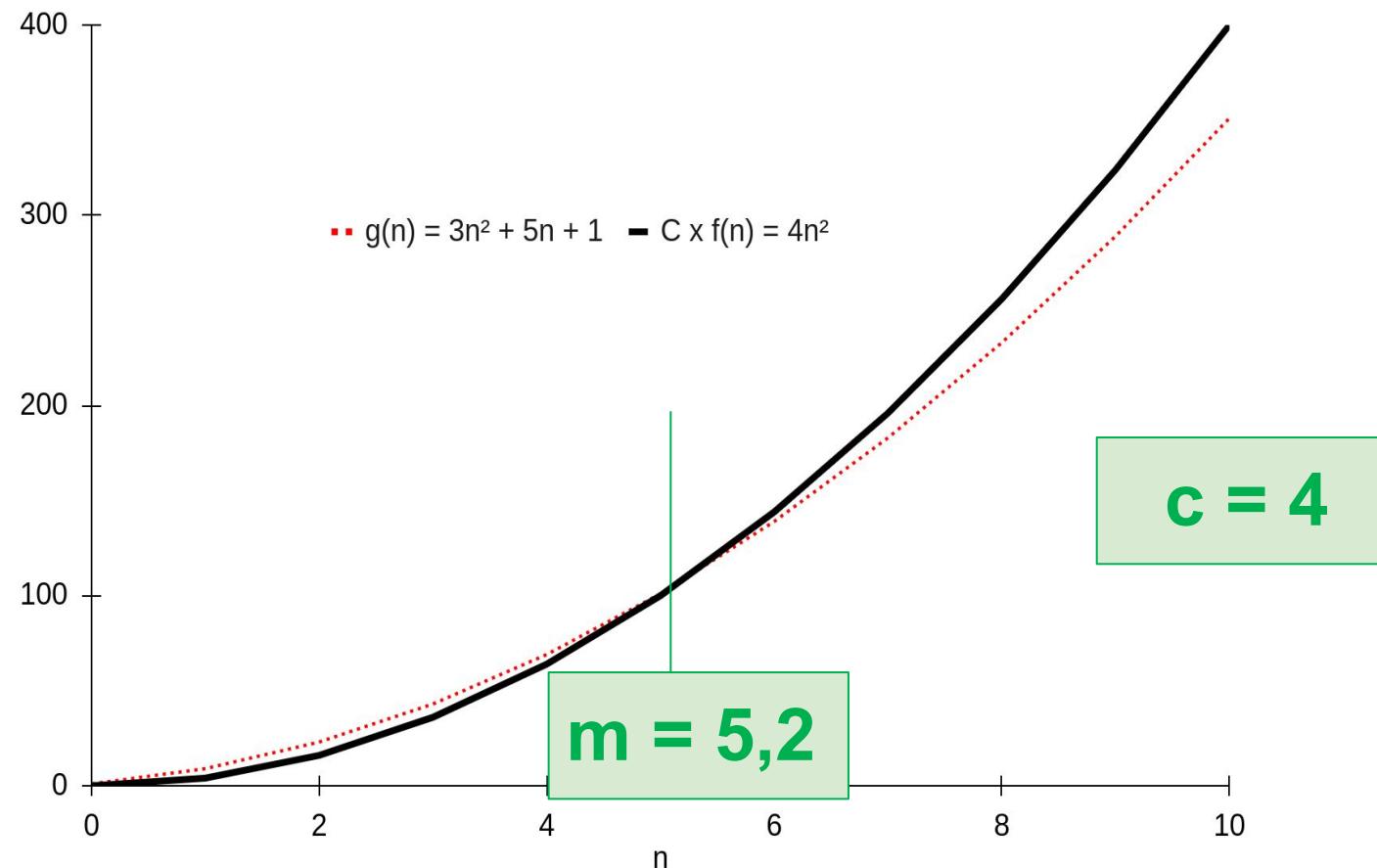


Exercício Resolvido (15)

• Dada a definição da notação O:



- a) Mostre os valores de c e m tal que, para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n^2|$, provando que $3n^2 + 5n + 1$ é $O(n^2)$

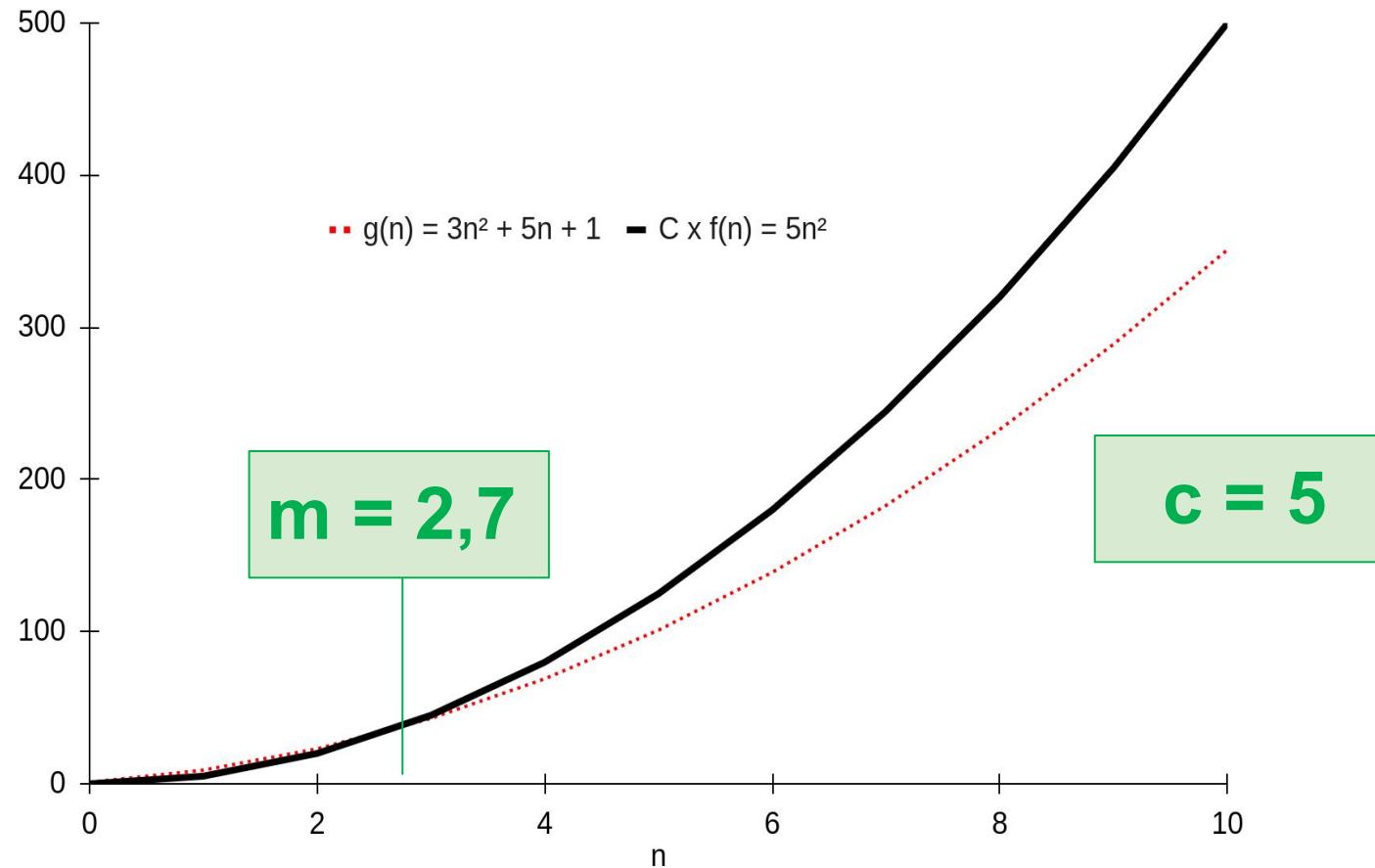


Exercício Resolvido (15)

• Dada a definição da notação O:



- a) Mostre os valores de c e m tal que, para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n^2|$, provando que $3n^2 + 5n + 1$ é $O(n^2)$



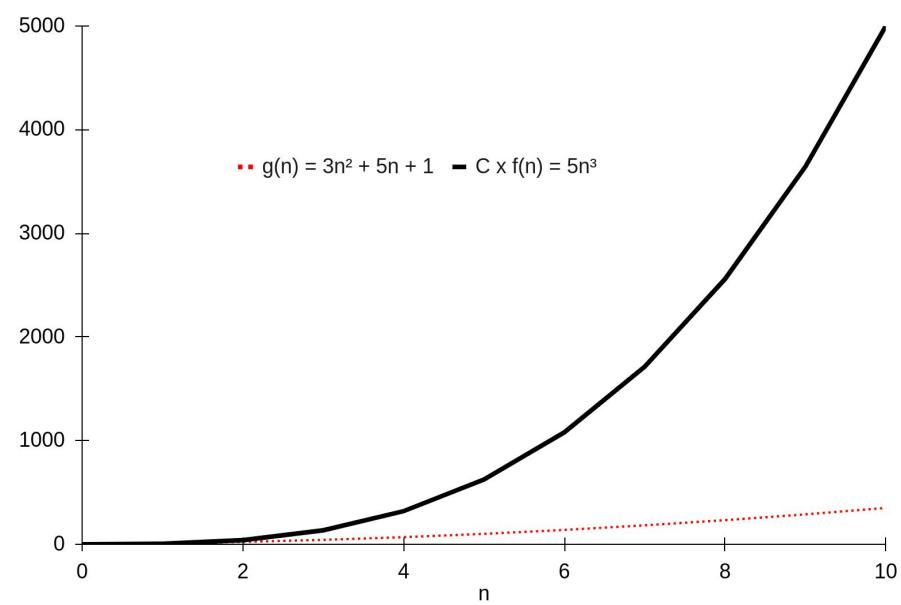
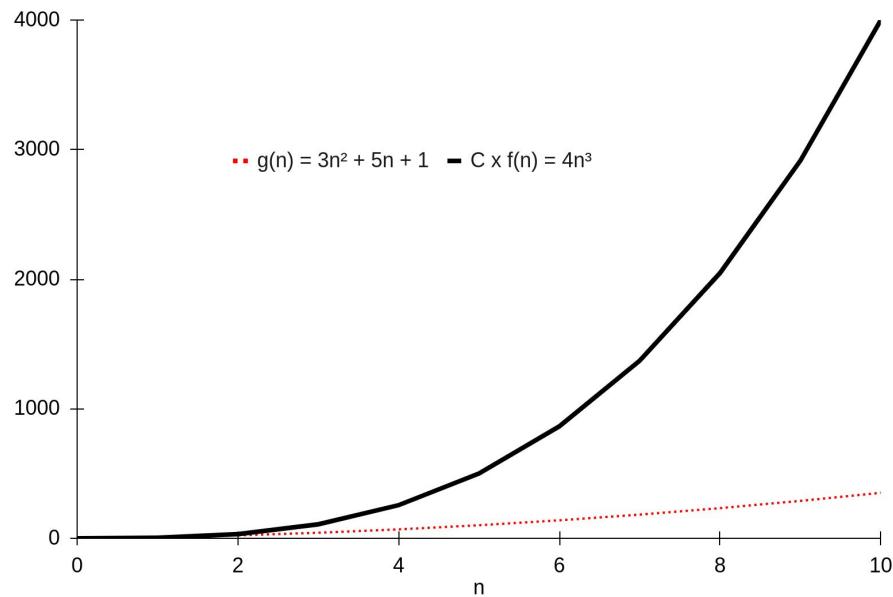
Exercício Resolvido (15)

• Dada a definição da notação O:



- b) Mostre os valores de c e m tal que, para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n^3|$, provando que $3n^2 + 5n + 1$ é $O(n^3)$

($c = 4$ e $m = 5,7$) e ($c = 5$ e $m = 2,7$) também atendem à letra (b)



Exercício Resolvido (15)

• Dada a definição da notação O:



c) Prove que $3n^2 + 5n + 1$ não é $O(n)$

Não existe par (c, m) tal que para $n \geq m$, $|3n^2 + 5n + 1| \leq c \times |n|$ seja verdadeira. Aumentando o valor de c , apenas retardamos o momento em que a curva quadrática supera a linear

Exercício Resolvido (15)



• Dada a definição da notação O:

c) Prove que $3n^2 + 5n + 1$ não é $O(n)$

c = 100

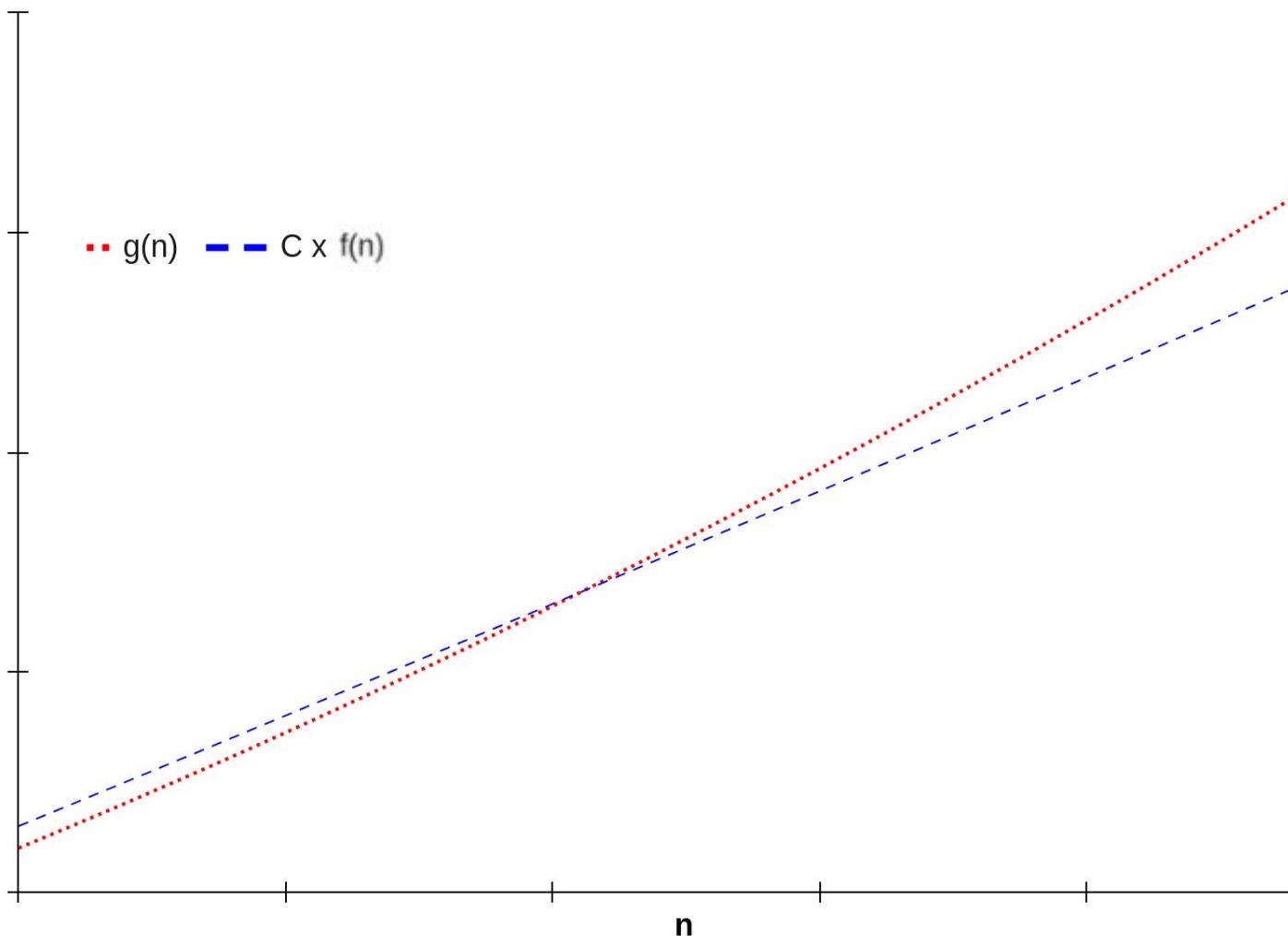
n	$g(n) = 3n^2 + 5n + 1$	$C \times f(n) = 100 \times n$
0	1	0
5	101	500
10	351	1000
15	751	1500
20	1301	2000
25	2001	2500
30	2851	3000
35	3851	3500
40	5001	4000
45	6301	4500
50	7751	5000

c = 1000

n	$g(n) = 3n^2 + 5n + 1$	$C \times f(n) = 1000 \times n$
0	1	0
50	7751	50000
100	30501	100000
150	68251	150000
200	121001	200000
250	188751	250000
300	271501	300000
350	369251	350000
400	482001	400000
450	609751	450000
500	752501	500000

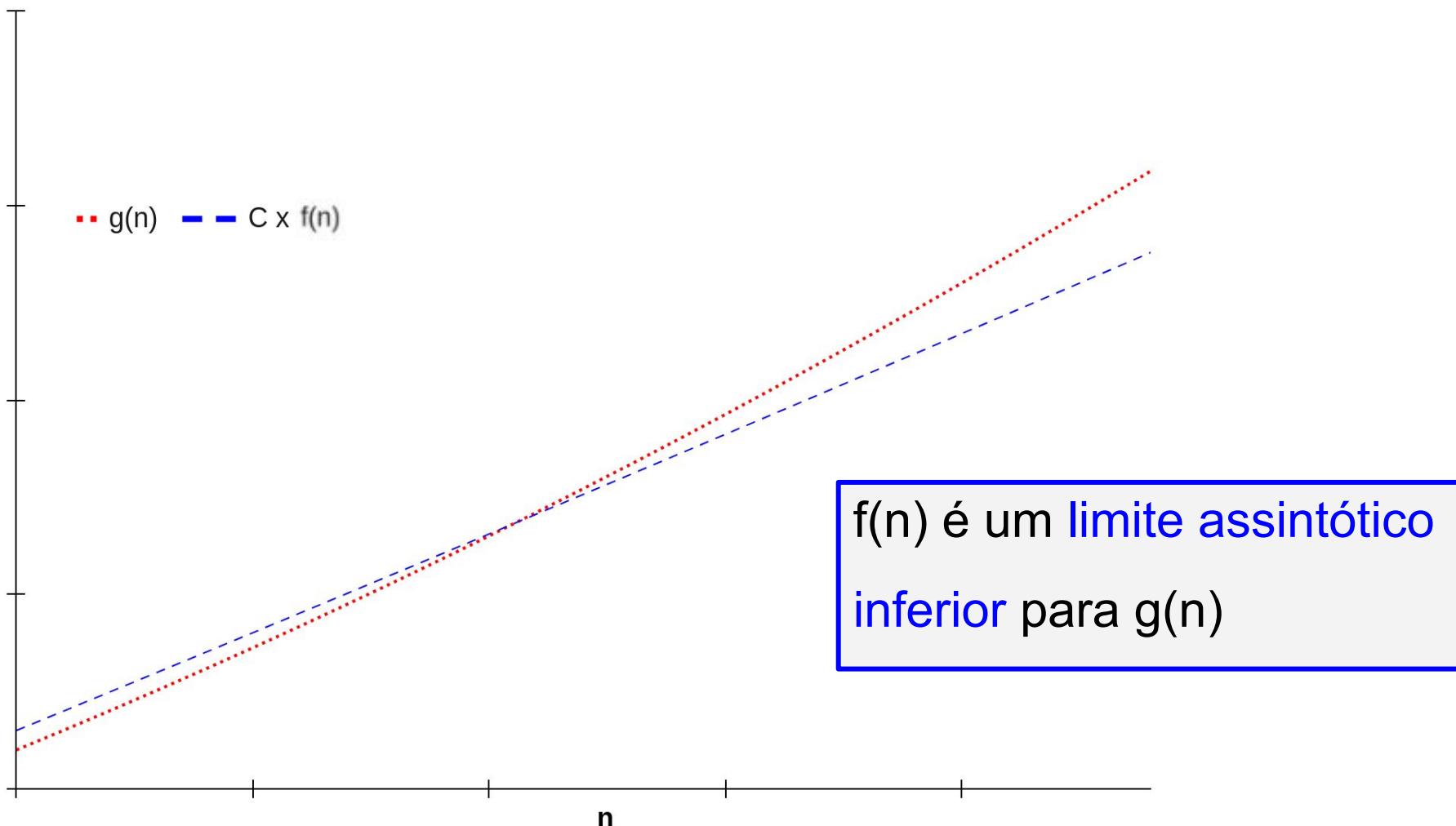
Definição da Notação Ω

- $g(n)$ é $\Omega(f(n))$, se existirem as constantes positivas c e m tais que, para $n \geq m$, temos que $|g(n)| \geq c \times |f(n)|$



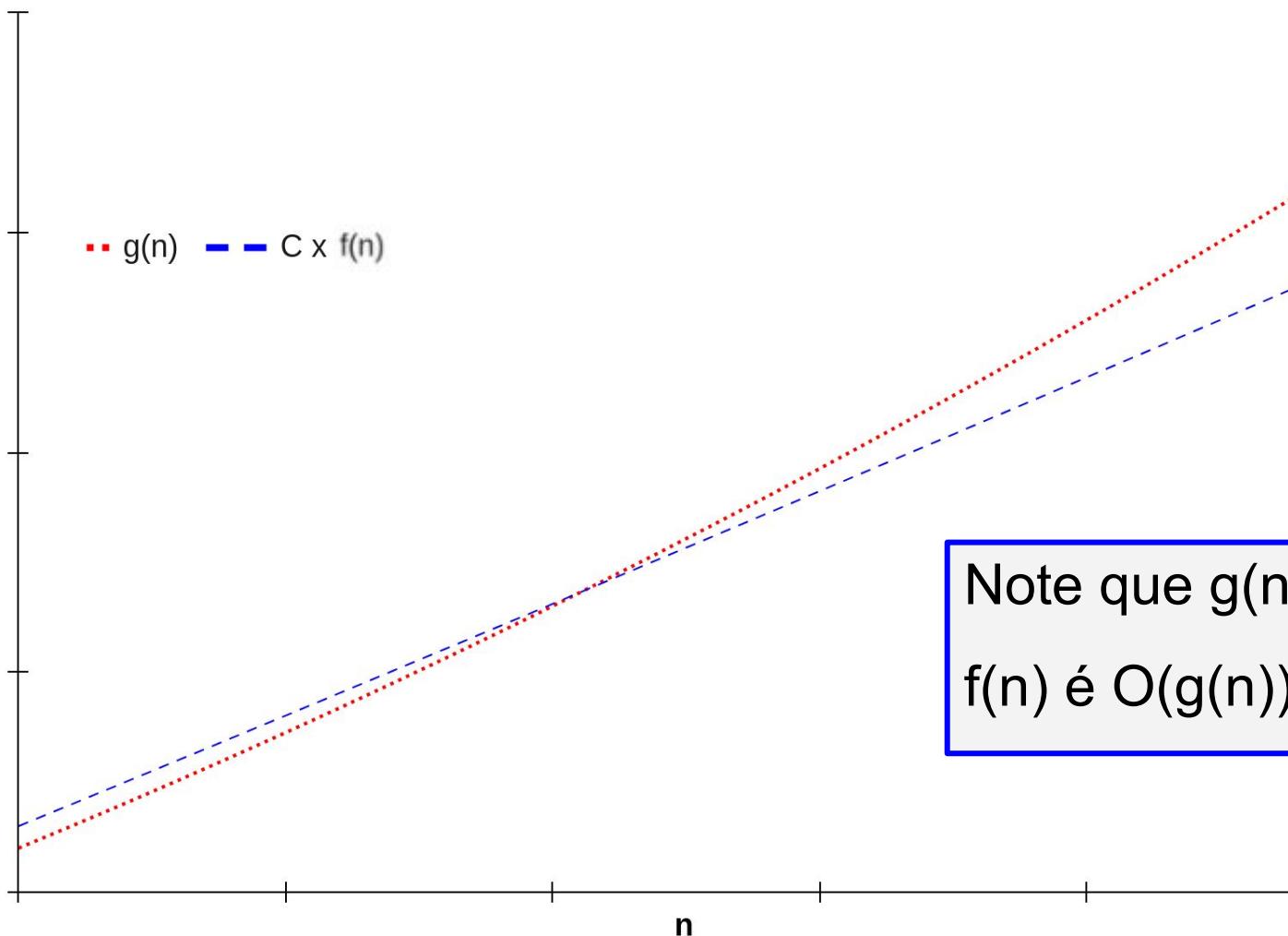
Definição da Notação Ω

- $g(n)$ é $\Omega(f(n))$, se existirem as constantes positivas c e m tais que, para $n \geq m$, temos que $|g(n)| \geq c \times |f(n)|$



Definição da Notação Ω

- $g(n)$ é $\Omega(f(n))$, se existirem as constantes positivas c e m tais que, para $n \geq m$, temos que $|g(n)| \geq c \times |f(n)|$



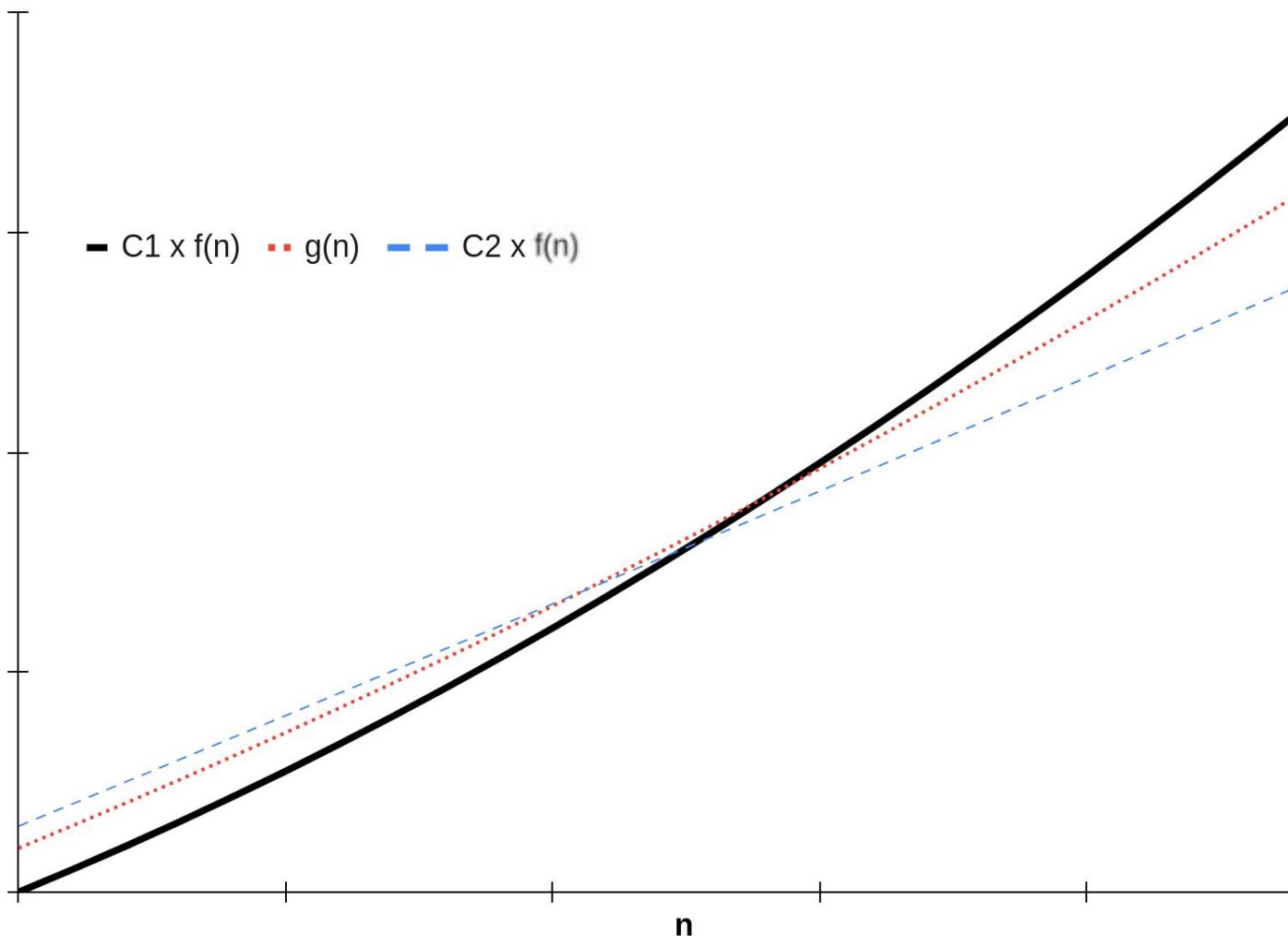
Exercício (6)

• Dada a definição da notação Ω :

- a) Mostre os valores de c e m tal que, para $n \geq m$, $|g(n)| \geq c \times |f(n)|$, provando que $3n^2 + 5n + 1$ é $\Omega(n^2)$
- b) Mostre os valores de c e m tal que, para $n \geq m$, $|g(n)| \geq c \times |f(n)|$, provando que $3n^2 + 5n + 1$ é $\Omega(n)$
- c) Prove que $3n^2 + 5n + 1$ não é $\Omega(n^3)$

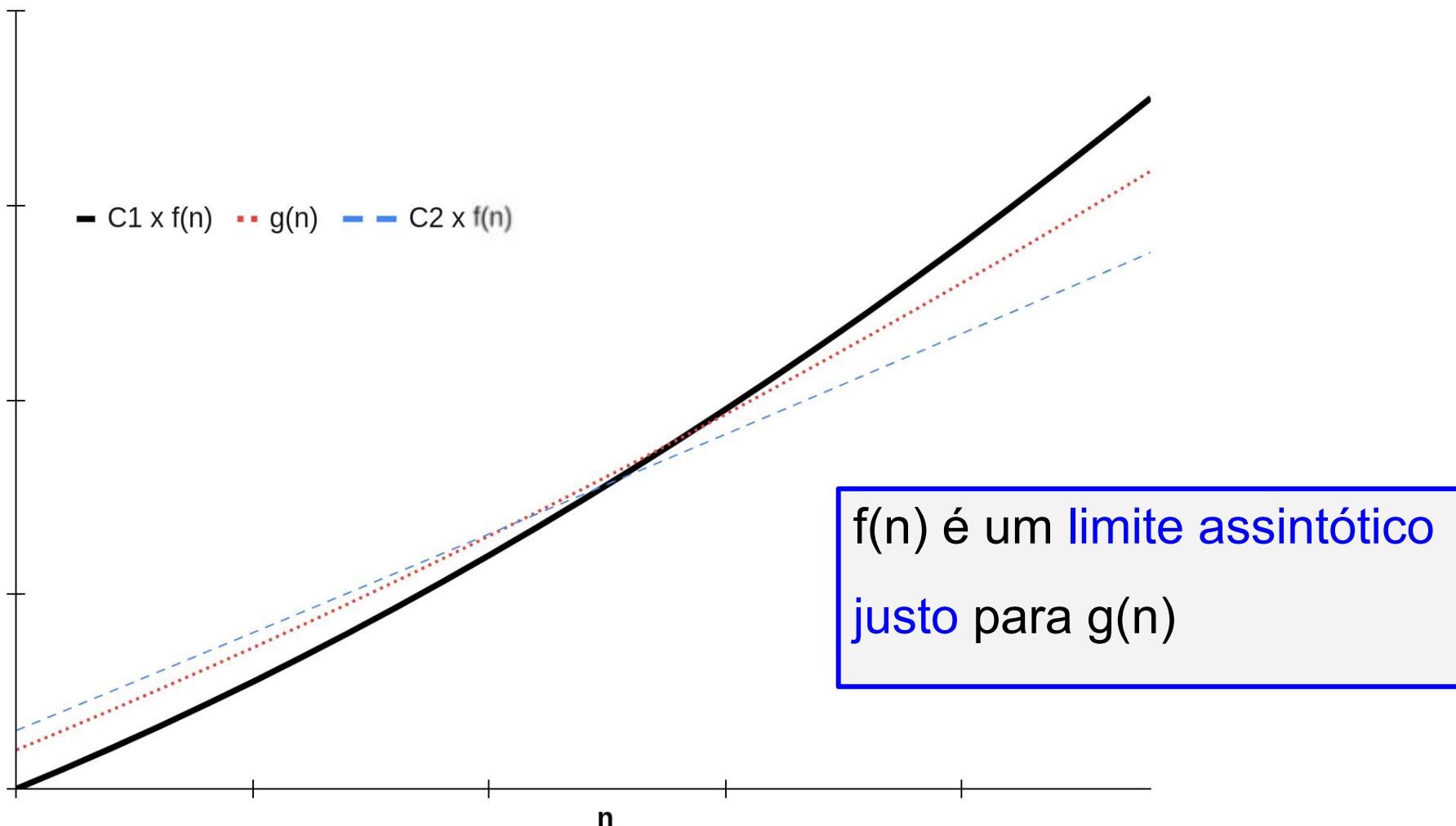
Definição da Notação Θ

- $g(n) = \Theta(f(n))$, se existirem constantes positivas c_1 , c_2 e m tais que, para $n \geq m$, temos que $c_1 \times |f(n)| \leq |g(n)| \leq c_2 \times |f(n)|$



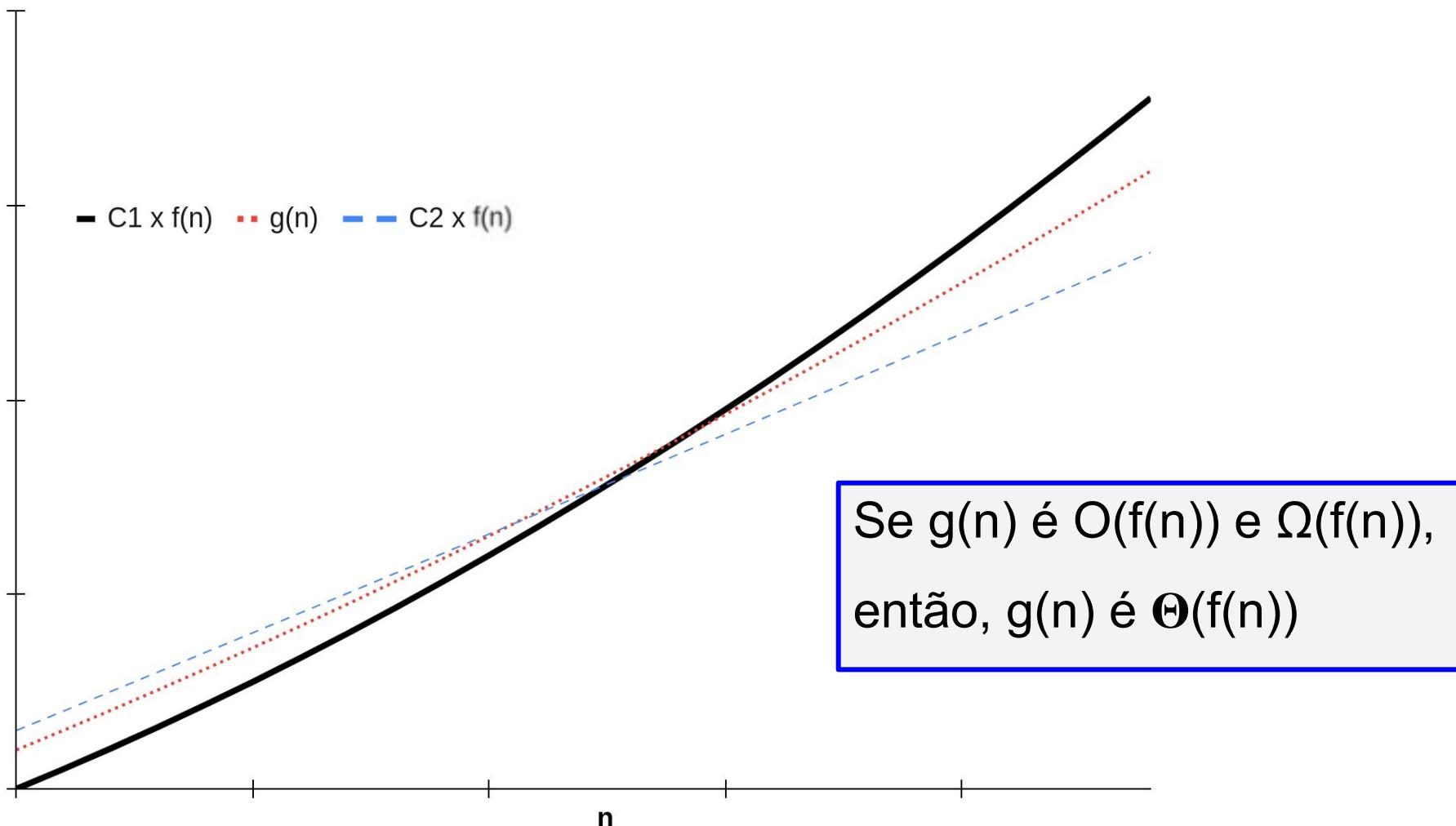
Definição da Notação Θ

- $g(n) = \Theta(f(n))$, se existirem constantes positivas c_1 , c_2 e m tais que, para $n \geq m$, temos que $c_1 \times |f(n)| \leq |g(n)| \leq c_2 \times |f(n)|$



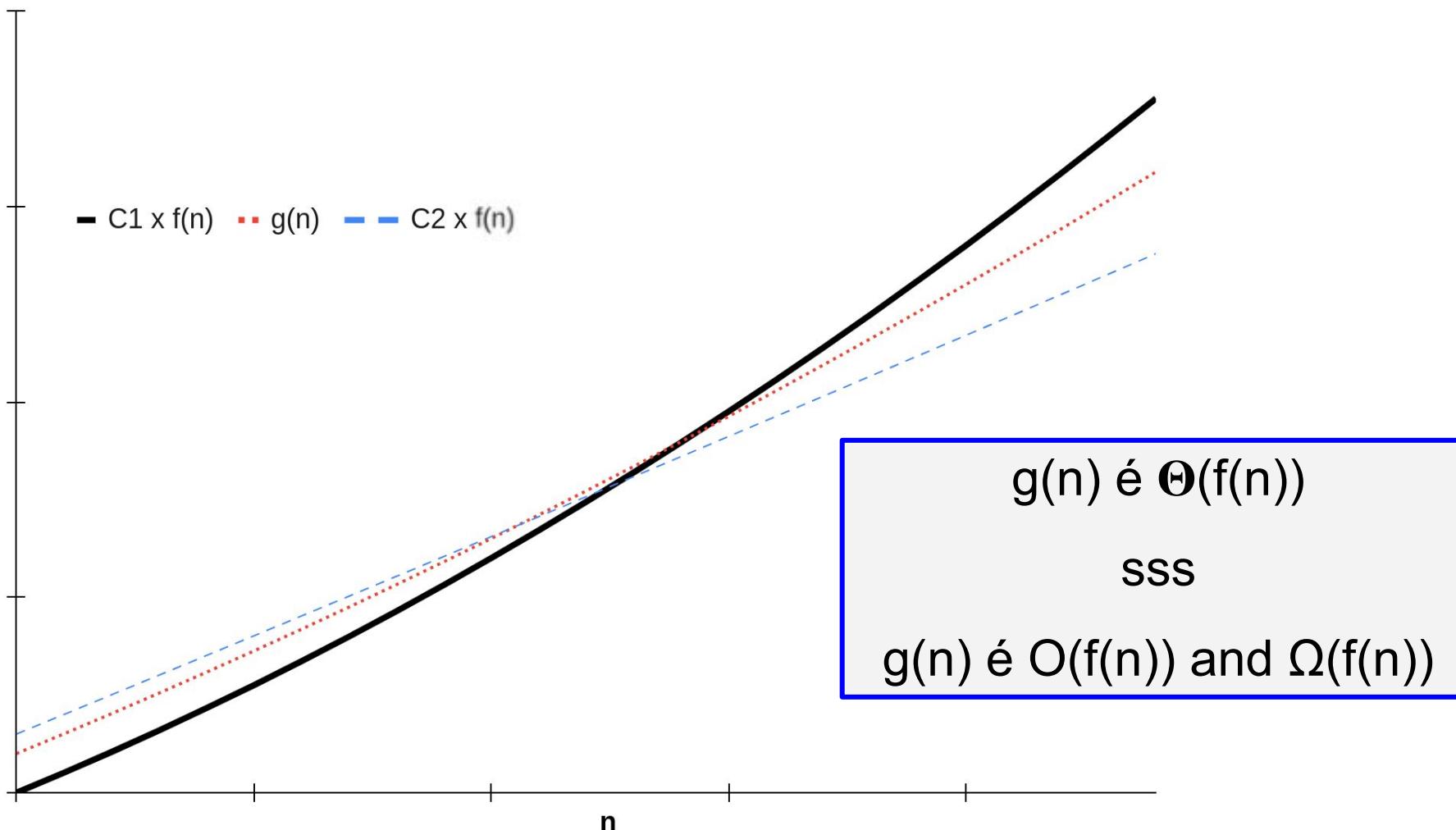
Definição da Notação Θ

- $g(n) = \Theta(f(n))$, se existirem constantes positivas c_1 , c_2 e m tais que, para $n \geq m$, temos que $c_1 \times |f(n)| \leq |g(n)| \leq c_2 \times |f(n)|$



Definição da Notação Θ

- $g(n) = \Theta(f(n))$, se existirem constantes positivas c_1 , c_2 e m tais que, para $n \geq m$, temos que $c_1 \times |f(n)| \leq |g(n)| \leq c_2 \times |f(n)|$



Exercício (7)

• Dada a definição da notação Θ :

- a) Mostre um valor para c_1 , c_2 e m tal que, para $n \geq m$, $c_1 \times |f(n)| \leq |g(n)| \leq c_2 \times |f(n)|$, provando que $3n^2 + 5n + 1$ é $\Theta(n^2)$
- b) Prove que $3n^2 + 5n + 1$ não é $\Theta(n)$
- c) Prove que $3n^2 + 5n + 1$ não é $\Theta(n^3)$

Classe de Algoritmos

- Constante: $O(1)$
- Logarítmico: $O(\lg n)$
- Linear: $O(n)$
- Linear-logarítmico: $O(n \lg n)$
- Quadrático: $O(n^2)$
- Cúbico: $O(n^3)$
- Exponencial: $O(c^n)$
- Fatorial: $O(n!)$

Algoritmos Polinomiais

- Um algoritmo é polinomial se é $O(n^p)$ para algum inteiro p
- Problemas com algoritmos polinomiais são considerados tratáveis
- Problemas para os quais não há algoritmos polinomiais são considerados intratáveis
- Classes de problemas e o problema $P = NP$?

Exercício (8)

- Faça um resumo sobre Teoria da Complexidade, Classes de Problemas P, NP e NP-Completo. Use LaTeX e siga o modelo de artigos da SBC (sem abstract, resumo nem seções) com no máximo duas página

Exercício Resolvido (16)

- Apresente a função e a complexidade para os números de comparações e movimentações de registros para o pior e melhor caso

```
void imprimirMaxMin(int [] array, int n){  
    int maximo, minimo;  
  
    if (array[0] > array[1]){  
        maximo = array[0];      minimo = array[1];  
    } else {  
        maximo = array[1];      minimo = array[0];  
    }  
  
    for (int i = 2; i < n; i++){  
        if (array[i] > maximo){  
            maximo = array[i];  
        } else if (array[i] < minimo){  
            minimo = array[i];  
        }  
    }  
}
```

Exercício Resolvido (16)

- Apresente a função e a complexidade para os números de comparações e movimentações de registros para o pior e melhor caso

```
void imprimirMaxMin(int f[ ] array, int n){
```

função de complexidade

	MOV	CMP
PIOR	$f(n) = 2 + (n - 2)$	$f(n) = 1 + 2(n - 2)$
MELHOR	$f(n) = 2 + (n - 2) \times 0$	$f(n) = 1 + (n - 2)$

complexidade

	MOV	CMP
PIOR	$O(n), \Omega(n) \text{ e } \Theta(n)$	$O(n), \Omega(n) \text{ e } \Theta(n)$
MELHOR	$O(1), \Omega(1) \text{ e } \Theta(1)$	$O(n), \Omega(n) \text{ e } \Theta(n)$

```
}
```



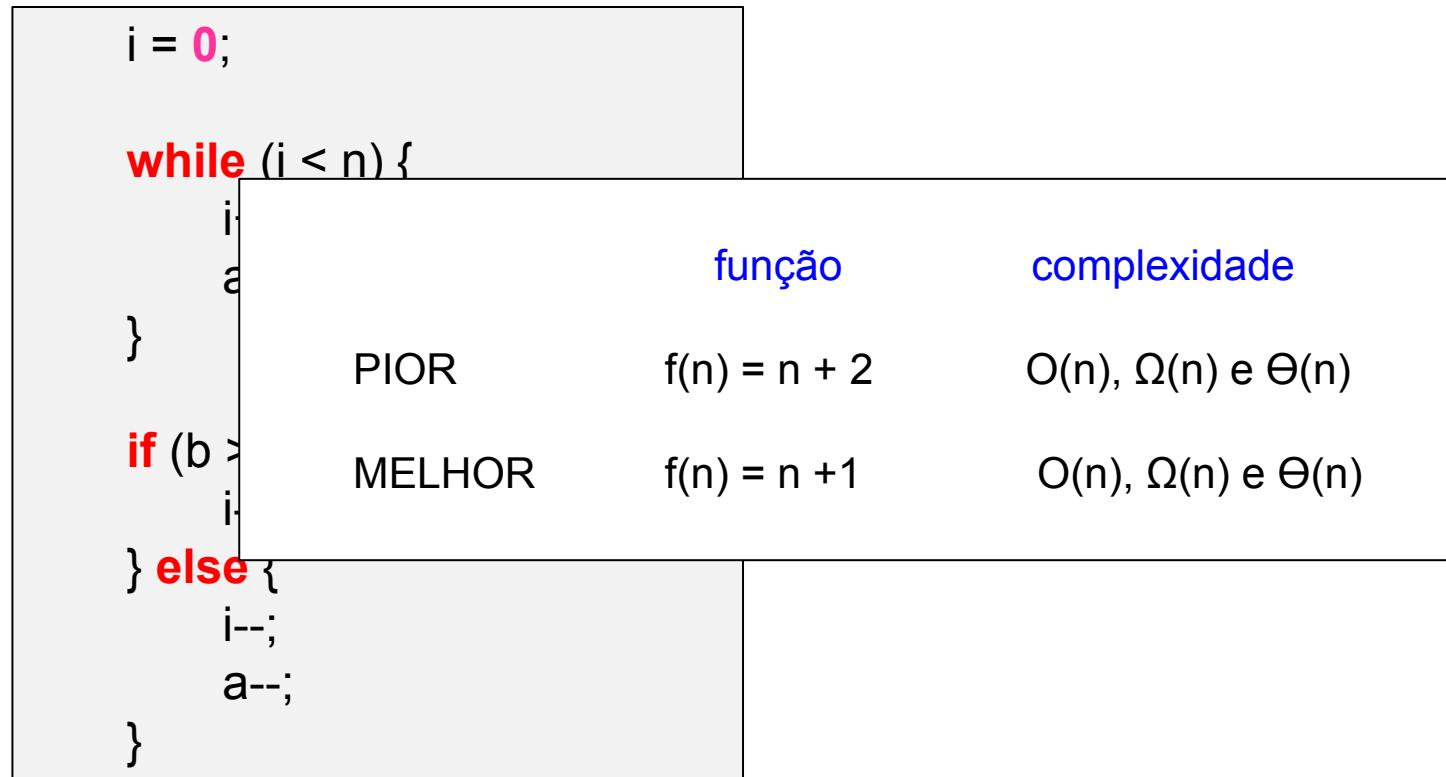
Exercício Resolvido (17)

- Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso

```
i = 0;  
  
while (i < n) {  
    i++;  
    a--;  
}  
  
if (b > c) {  
    i--;  
} else {  
    i--;  
    a--;  
}
```

Exercício Resolvido (17)

- Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso



Exercício Resolvido (18)

- Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso

```
for (i = 0; i < n; i++) {  
    for (j = 0; j < n; j++) {  
        a--;  
        b--;  
    }  
    c--;  
}
```

Exercício Resolvido (18)

- Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso

```
for (i = 0; i < n; i++) {  
    for (j = 0; j < n; j++) {  
        ...  
    }  
}
```



	função	complexidade
TODOS	$f(n) = (2n + 1)n$	$O(n^2)$, $\Omega(n^2)$ e $\Theta(n^2)$

Exercício Resolvido (19)

- Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso

```
for (i = 0; i < n; i++) {  
    for (j = 1; j <= n; j *= 2) {  
        b--;  
    }  
}
```

Exercício Resolvido (19)

- Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso

```
for (i = 0; i < n; i++) {  
    for (j = 1; j <= n; j *= 2) {  
        b .
```



função

complexidade

TODOS

$$f(n) = (\lg(n) + 1) * n = n * \lg(n) + n$$

 $O(n \times \lg(n))$, $\Omega(n \times \lg(n))$ e $\Theta(n \times \lg(n))$

Exercício (9)

- Suponha um sistema de monitoramento contendo os métodos telefone, luz, alarme, sensor e câmera, apresente a função e ordem de complexidade para o pior e melhor caso: (a) método alarme; (b) outros métodos.

```
void sistemaMonitoramento() {
    if (telefone() == true && luz() == true){
        alarme(0);
    } else {
        alarme(1);
    }
    for (int i = 2; i < n; i++){
        if (sensor(i- 2) == true){
            alarme (i - 2);
        } else if (camera(i- 2) == true){
            alarme (i - 2 + n);
        }
    }
}
```

Exercício (10)

- Apresente um código, defina duas operações relevantes e apresente a função e a complexidade para as operações escolhidas no pior e melhor caso

Exercício Resolvido (20)

- Apresente o tipo de crescimento que melhor caracteriza as funções abaixo
(Khan Academy, adaptado)

	Constante	Linear	Polinomial	Exponencial
$3n$				
1				
$(3/2)n$				
$2n^3$				
2^n				
$3n^2$				
1000				
$(3/2)^n$				

Exercício Resolvido (20)

- Apresente o tipo de crescimento que melhor caracteriza as funções abaixo
(Khan Academy, adaptado)

	Constante	Linear	Polinomial	Exponencial
$3n$		✓		
1	✓			
$(3/2)n$		✓		
$2n^3$			✓	
2^n				✓
$3n^2$			✓	
1000	✓			
$(3/2)^n$				✓

Exercício Resolvido (21)

- Classifique as funções $f_1(n) = n^2$, $f_2(n) = n$, $f_3(n) = 2^n$, $f_4(n) = (3/2)^n$, $f_5(n) = n^3$ e $f_6(n) = 1$ de acordo com o crescimento, do mais lento para o mais rápido (Khan Academy, adaptado)

Exercício Resolvido (21)

- Classifique as funções $f_1(n) = n^2$, $f_2(n) = n$, $f_3(n) = 2^n$, $f_4(n) = (3/2)^n$, $f_5(n) = n^3$ e $f_6(n) = 1$ de acordo com o crescimento, do mais lento para o mais rápido (Khan Academy, adaptado)

$$f_6(n) = 1$$



$$f_2(n) = n$$

$$f_1(n) = n^2$$

$$f_5(n) = n^3$$

$$f_4(n) = (3/2)^n$$

$$f_3(n) = 2^n$$

Exercício Resolvido (22)

- Classifique as funções $f_1(n) = n \cdot \log_6(n)$, $f_2(n) = \lg(n)$, $f_3(n) = \log_8(n)$, $f_4(n) = 8n^2$, $f_5(n) = n \cdot \lg(n)$, $f_6(n) = 64$, $f_7(n) = 6n^3$, $f_8(n) = 8^{2n}$ e $f_9(n) = 4n$ de acordo com o crescimento, do mais lento para o mais rápido (Khan Academy, adaptado)

Exercício Resolvido (22)

- Classifique as funções $f_1(n) = n \cdot \log_6(n)$, $f_2(n) = \lg(n)$, $f_3(n) = \log_8(n)$, $f_4(n) = 8n^2$, $f_5(n) = n \cdot \lg(n)$, $f_6(n) = 64$, $f_7(n) = 6n^3$, $f_8(n) = 8^{2n}$ e $f_9(n) = 4n$ de acordo com o crescimento, do mais lento para o mais rápido (Khan Academy, adaptado)

$$f_6(n) = 64$$

$$f_3(n) = \log_8(n)$$

$$f_2(n) = \lg(n)$$

$$f_9(n) = 4n$$

$$f_1(n) = n \cdot \log_6(n)$$

$$f_5(n) = n \cdot \lg(n)$$

$$f_4(n) = 8n^2$$

$$f_7(n) = 6n^3$$

$$f_8(n) = 8^{2n}$$



Exercício Resolvido (23)

- Faça a correspondência entre cada função $f(n)$ com sua $g(n)$ equivalente, em termos de Θ . Essa correspondência acontece quando $f(n) = \Theta(g(n))$ (Khan Academy, adaptado)

$f(n)$	$g(n)$
$n + 30$	n^4
$n^2 + 2n - 10$	$3n - 1$
$n^3 \cdot 3n$	$\lg(2n)$
$\lg(n)$	$n^2 + 3n$

Exercício Resolvido (23)

- Faça a correspondência entre cada função $f(n)$ com sua $g(n)$ equivalente, em termos de Θ . Essa correspondência acontece quando $f(n) = \Theta(g(n))$ (Khan Academy, adaptado)

$f(n)$	$g(n)$
$n + 30$	n^4
$n^2 + 2n - 10$	$3n - 1$
$n^3 \cdot 3n$	$\lg(2n)$
$\lg(n)$	$n^2 + 3n$

Exercício (11)

- No Exercício Resolvido (10), verificamos que quando desejamos pesquisar a existência de *um* elemento em um *array* de números reais é adequado executar uma pesquisa sequencial cujo custo é $\Theta(n)$. Nesse caso, o custo de ordenar o *array* e, em seguida, aplicar uma pesquisa binária é mais elevado, $\Theta(n * \lg(n)) + \Theta(\lg(n)) = \Theta(n * \lg(n))$. Agora, supondo que desejamos efetuar *n* pesquisas, responda qual das duas soluções é mais eficiente