Análise de Complexidade

Parte 2

Prof. Kennedy Reurison Lopes

July 4, 2023

Cálculo da complexidade

Tempo de Execução:

$$T(n) = \sum_{i=1}^{n} t_i n_i$$

Cálculo da complexidade

Tempo de Execução:

$$T(n) = \sum_{i=1}^{n} t_i n_i$$

- *i* índice da instrução;

Cálculo da complexidade

Tempo de Execução:

$$T(n) = \sum_{i=1}^{n} t_i n_i$$

- i índice da instrução;
- t_i o tempo necessário para a excução da instrução i;

Cálculo da complexidade

Tempo de Execução:

$$T(n) = \sum_{i=1}^{n} t_i n_i$$

- i índice da instrução;
- t_i o tempo necessário para a excução da instrução i;
- n_i o número de vezes que a instrução i é executada.

Obtenção de t_i

O cálculo de t_i depende:

Obtenção de t_i

O cálculo de t_i depende:

- Memória disponível do computador;

Obtenção de t_i

O cálculo de t_i depende:

- Memória disponível do computador;
- Desempenho do processador;

Obtenção de t_i

O cálculo de t_i depende:

- Memória disponível do computador;
- Desempenho do processador;
- Arquitetura e estado dos dispositivos naquele momento de execução.

Obtenção de t_i

O cálculo de t_i depende:

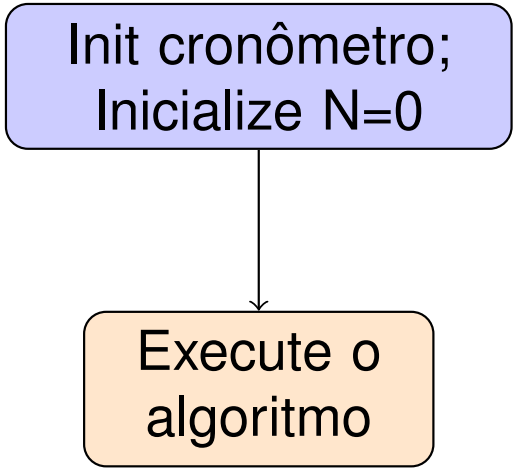
- Memória disponível do computador;
- Desempenho do processador;
- Arquitetura e estado dos dispositivos naquele momento de execução.

Até mesmo o mesmo computador tem tempo de respostas diferentes.

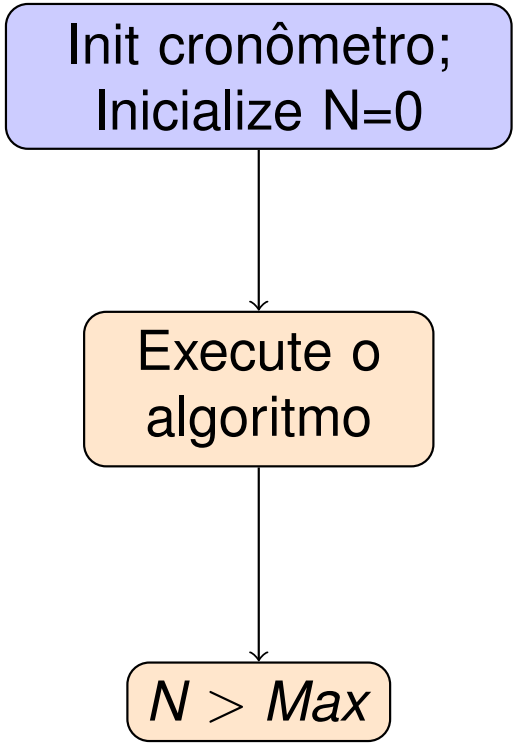
Experimento para cálculo do tempo

Init cronômetro;
Inicialize N=0

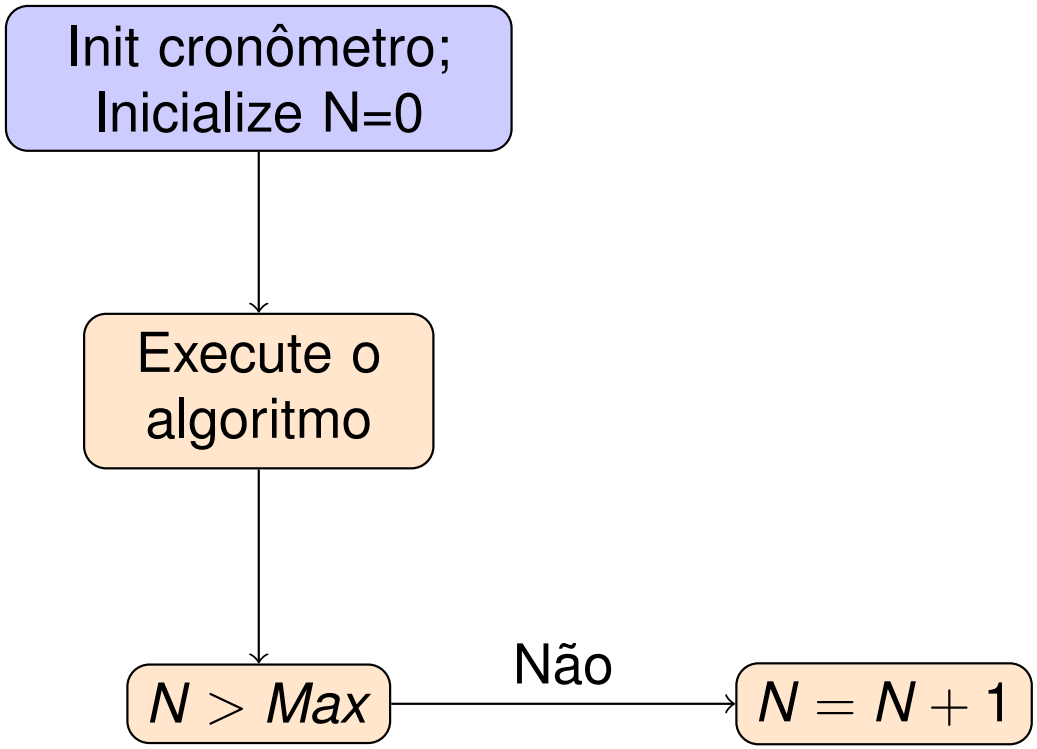
Experimento para cálculo do tempo



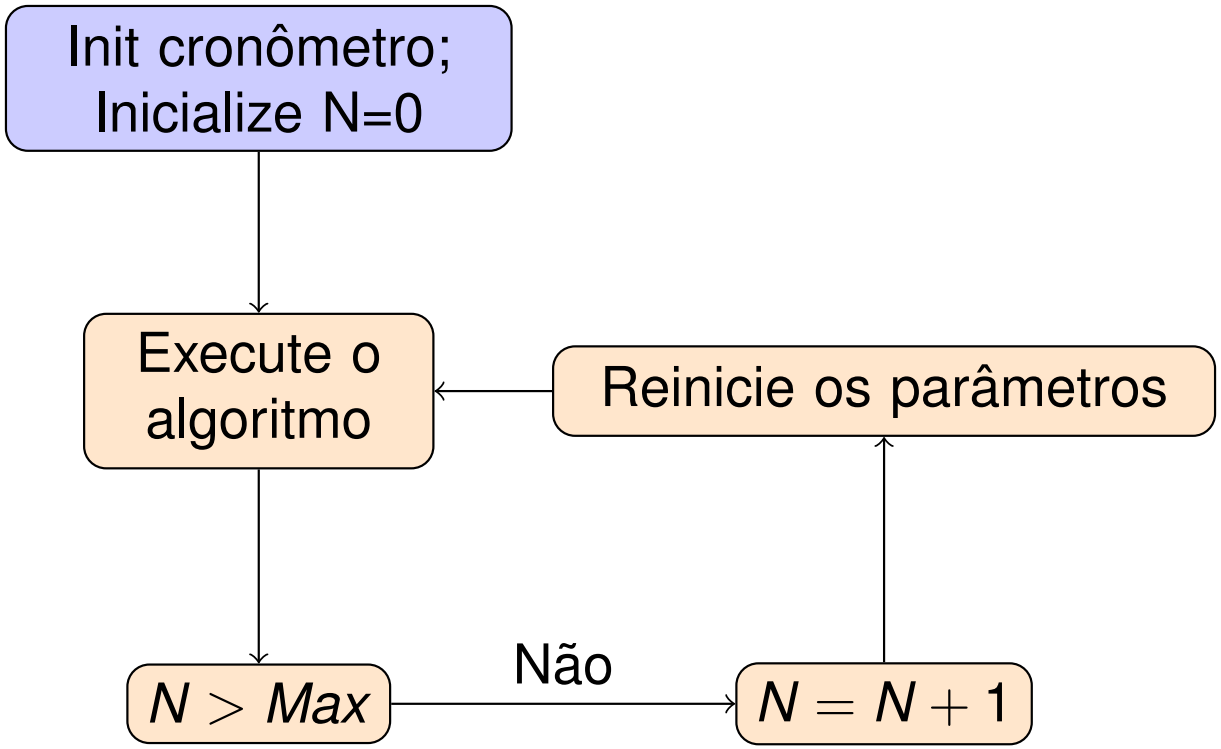
Experimento para cálculo do tempo



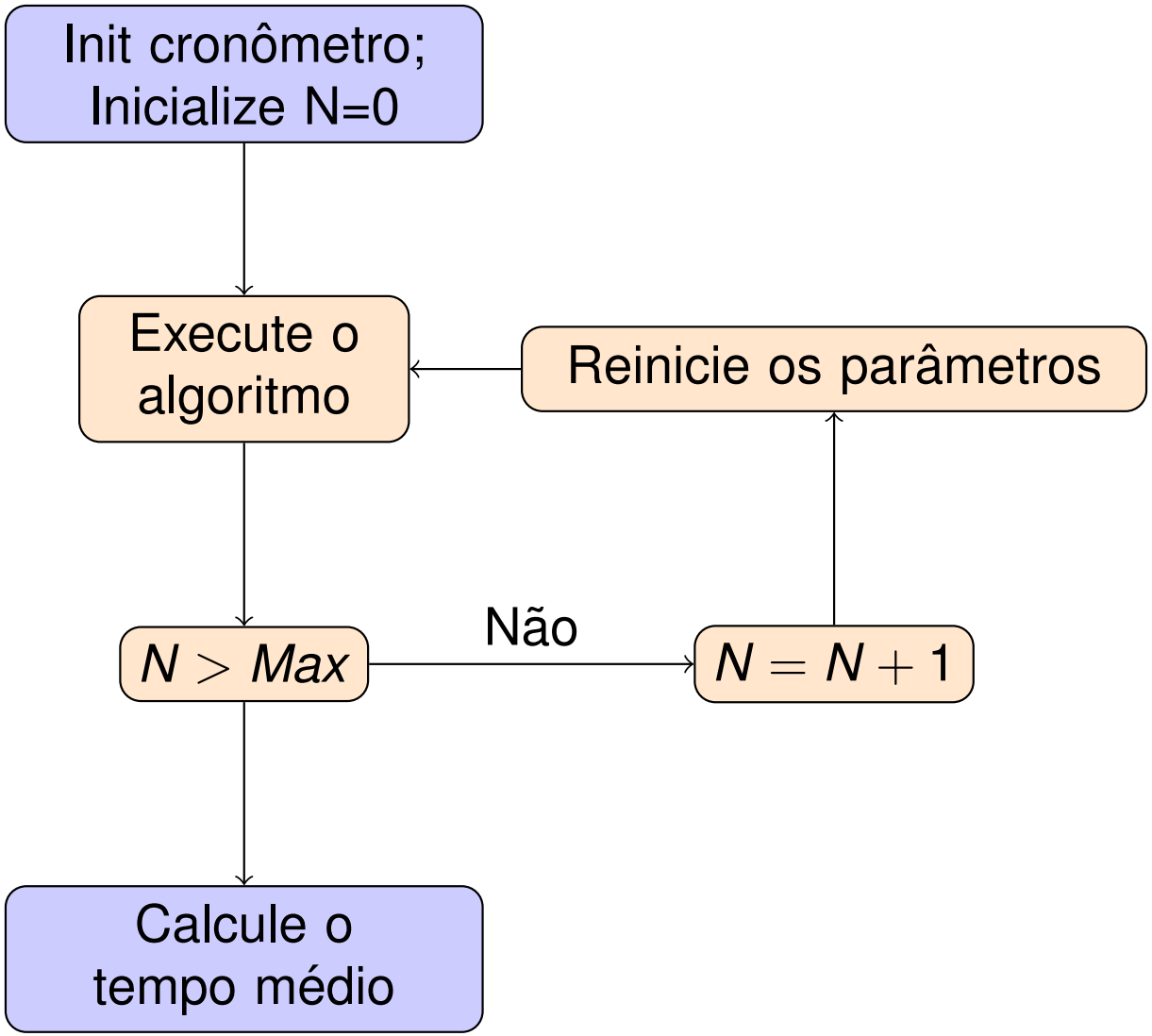
Experimento para cálculo do tempo



Experimento para cálculo do tempo



Experimento para cálculo do tempo



Contagem de Frequência - Algoritmo 1

```
1  int main() {
2      int x = 0;
3      x = x + 1;
4      return 0;
5  }
```

Contagem de Frequência - Algoritmo 2

```
1  int main() {
2      int x = 0, n=10;
3      for (int i = 0; i < n; i++) {
4          x = x + 1;
5      }
6      return 0;
7  }
```

Contagem de frequência

Exemplo: Conte as operações (contagem de frequência) realizadas na operação de uma soma geométrica definida por:

$$S = \sum_{i=0}^{n} x^i$$

Analise as instruções do algoritmo que o descreve logo a seguir:

Contagem de frequência - Algoritmo 4

Contagem de frequência - Algoritmo 4

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

L2) 1

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

L2) 1

L3) $n + 2$

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

L2) 1

L3) $n+2$

L4) $n+1$

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

L2) 1

L3) $n + 2$

L4) $n + 1$

L6) $\sum_{i=0}^{n} i$

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

L2) 1

L3) $n+2$

L4) $n+1$

L6) $\sum_{i=0}^{n} i$

L7) $n+1$

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

L2) 1

L3) $n+2$

L4) $n+1$

L6) $\sum_{i=0}^{n} i$

L7) $n+1$

L9) 1

Contagem de frequência - Algoritmo 4

```
1  float soma(float x, int n) {
2      int soma = 0;
3      int i=0, j=0;
4      for (i = 0; i <= n; i++) {
5          int prod = 1;
6          for (j = 0; j < i; j++)
7              prod = prod * x;
8          soma + soma + prod;
9      }
10     return soma;
11 }
```

L2) 1

L3) $n+2$

L4) $n+1$

L6) $\sum_{i=0}^{n} i$

L7) $n+1$

L9) 1

Somando todos os tempos, temos o tempo total:

$$T(n) = \frac{n^2}{2} + \frac{7n}{2} + 6$$

Fórmula de *Horner*

Pode-se modificar o algoritmo para melhorar o tempo de execução utilizando o algoritmo de *Horner*.

$$
\begin{aligned}
S = \sum_{i=0}^{n} x^i &= 1 + x + x^2 + \ldots + x^n \\
&= 1 + x(1 + x + x^2 + \ldots + x^{n-1}) \\
&= 1 + x(1 + x(1 + x + x^2 + \ldots + x^{n-2})) \\
&= 1 + x(1 + x(1 + x(1 + \ldots (1 + x(1 + x))) \ldots))
\end{aligned}
$$

Contagem de frequência - Algoritmo 5

Contagem de frequência - Algoritmo 5

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

L2) 1

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

L2) 1

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

L2) 1

L3) n+2

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

L2) 1

L3) n+2

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

L2) 1

L3) n+2

L4) n+1

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

L2) 1

L3) n+2

L4) n+1

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

L2) 1

L3) n+2

L4) n+1

L6) 1

Contagem de frequência - Algoritmo 5

```
1 float horner(float x, int n) {
2     int i=0, soma = 0;
3     for (i = 0; i <= n; i++) {
4         soma = soma * x + 1;
5     }
6     return soma;
7 }
```

L2) 1
L3) n+2
L4) n+1
L6) 1

Portanto: $T(n) = 1 + (n+2) + (n+1) + 1 = 2n + 5$.

Fórmula fechada

Uma terceira solução é encontrar a fórmula fechada:

$$S = \sum_{i=0}^{n} i = 1 + x + x^2 + \ldots + x^n$$

Fórmula fechada

Uma terceira solução é encontrar a fórmula fechada:

$$S = \sum_{i=0}^{n} i = 1 + x + x^2 + \ldots + x^n$$

$$xS = x(1 + x + x^2 + \ldots + x^n)$$

Fórmula fechada

Uma terceira solução é encontrar a fórmula fechada:

$$S = \sum_{i=0}^{n} i = 1 + x + x^2 + \ldots + x^n$$

$$xS = x(1 + x + x^2 + \ldots + x^n)$$

$$xS = x + x^2 + \ldots + x^{n+1}$$

Fórmula fechada

Uma terceira solução é encontrar a fórmula fechada:

$$S = \sum_{i=0}^{n} i = 1 + x + x^2 + \ldots + x^n$$

$$xS = x(1 + x + x^2 + \ldots + x^n)$$

$$xS = x + x^2 + \ldots + x^{n+1}$$

$$xS + 1 = 1 + x + x^2 + \ldots + x^{n+1}$$

Fórmula fechada

Uma terceira solução é encontrar a fórmula fechada:

$$
\begin{aligned}
S = \sum_{i=0}^{n} i &= 1 + x + x^2 + \ldots + x^n \\
xS &= x(1 + x + x^2 + \ldots + x^n) \\
xS &= x + x^2 + \ldots + x^{n+1} \\
xS + 1 &= 1 + x + x^2 + \ldots + x^{n+1} \\
xS + 1 &= S + x^{n+1}
\end{aligned}
$$

Fórmula fechada

Uma terceira solução é encontrar a fórmula fechada:

$$
\begin{aligned}
S = \sum_{i=0}^{n} i &= 1 + x + x^2 + \ldots + x^n \\
xS &= x(1 + x + x^2 + \ldots + x^n) \\
xS &= x + x^2 + \ldots + x^{n+1} \\
xS + 1 &= 1 + x + x^2 + \ldots + x^{n+1} \\
xS + 1 &= S + x^{n+1} \\
xS - S &= x^{n+1} - 1
\end{aligned}
$$

Fórmula fechada

Uma terceira solução é encontrar a fórmula fechada:

$$
\begin{aligned}
S = \sum_{i=0}^{n} i &= 1 + x + x^2 + \ldots + x^n \\
xS &= x(1 + x + x^2 + \ldots + x^n) \\
xS &= x + x^2 + \ldots + x^{n+1} \\
xS + 1 &= 1 + x + x^2 + \ldots + x^{n+1} \\
xS + 1 &= S + x^{n+1} \\
xS - S &= x^{n+1} - 1 \\
(x-1)S &= x^{n+1} - 1
\end{aligned}
$$

Fórmula fechada

Uma terceira solução é encontrar a fórmula fechada:

$$
\begin{aligned}
S = \sum_{i=0}^{n} i &= 1 + x + x^2 + \ldots + x^n \\
xS &= x(1 + x + x^2 + \ldots + x^n) \\
xS &= x + x^2 + \ldots + x^{n+1} \\
xS + 1 &= 1 + x + x^2 + \ldots + x^{n+1} \\
xS + 1 &= S + x^{n+1} \\
xS - S &= x^{n+1} - 1 \\
(x-1)S &= x^{n+1} - 1 \\
S &= \frac{x^{n+1} - 1}{x - 1}
\end{aligned}
$$

Contagem de frequência - Algoritmo 6

Qual a complexidade desse algoritmo com a fórmula fechada?

Contagem de frequência - Algoritmo 6

Qual a complexidade desse algoritmo com a fórmula fechada?

```c
#include <math.h>
float soma(int x, int n) {
    return pow(x, n + 1) / (x - 1);
}
```

Contagem de frequência - Algoritmo 6

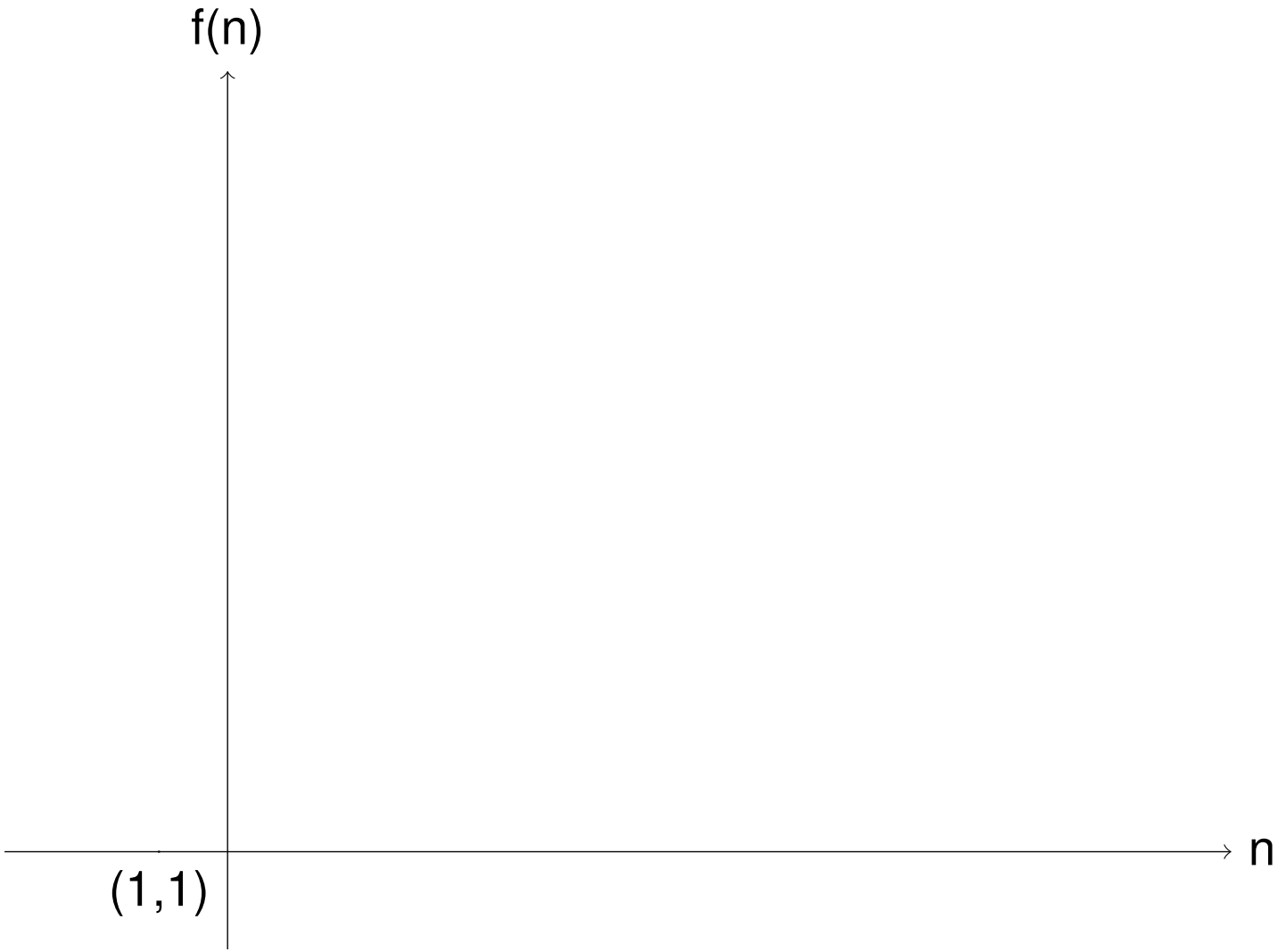
Qual a complexidade desse algoritmo com a fórmula fechada?

```
1 #include <math.h>
2 float soma(int x, int n) {
3     return pow(x, n + 1) / (x - 1);
4 }
```

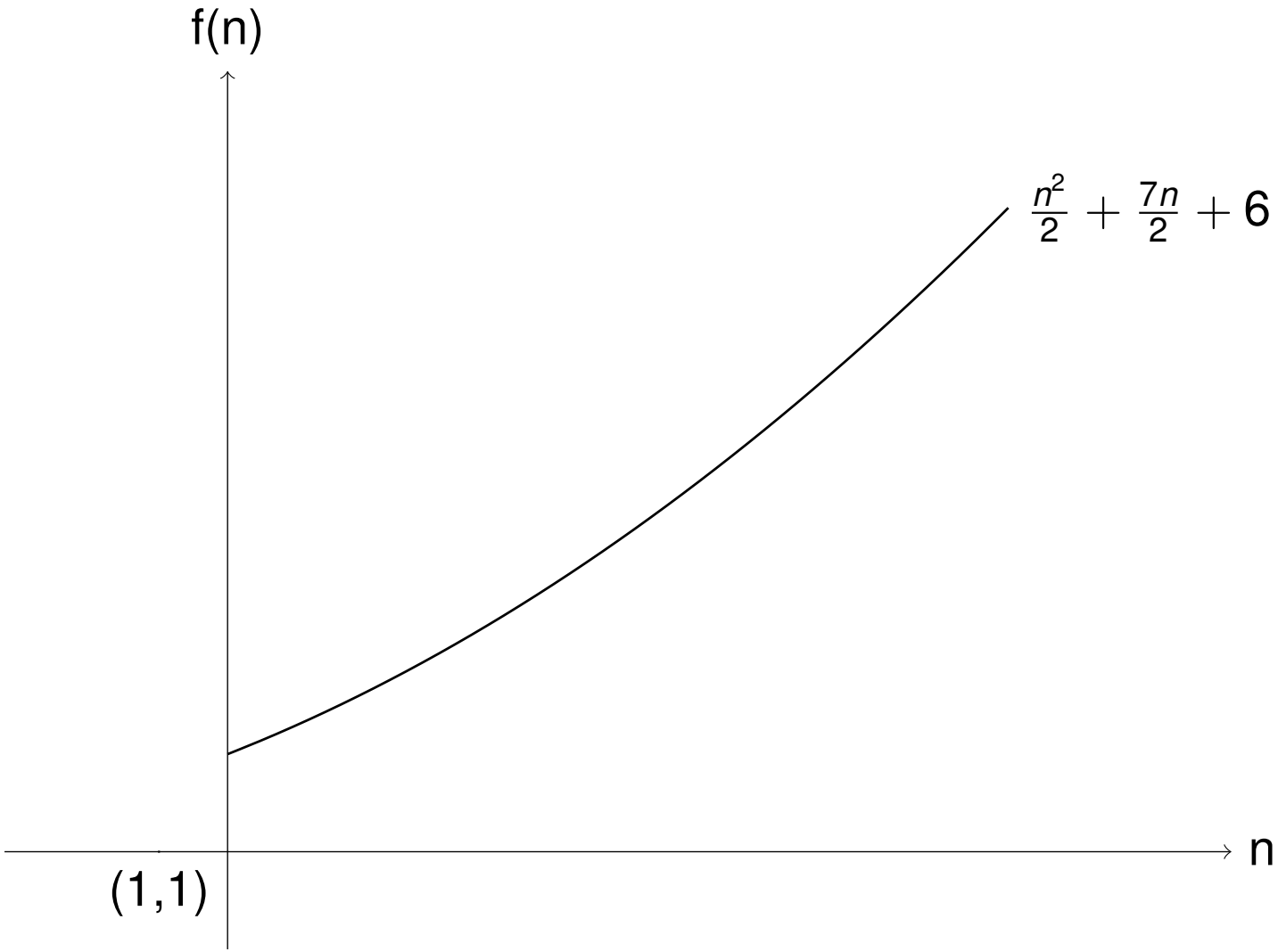
Considerando que *pow* tem complexidade logaritmica:

$$T(n) = log_2(n+1)$$

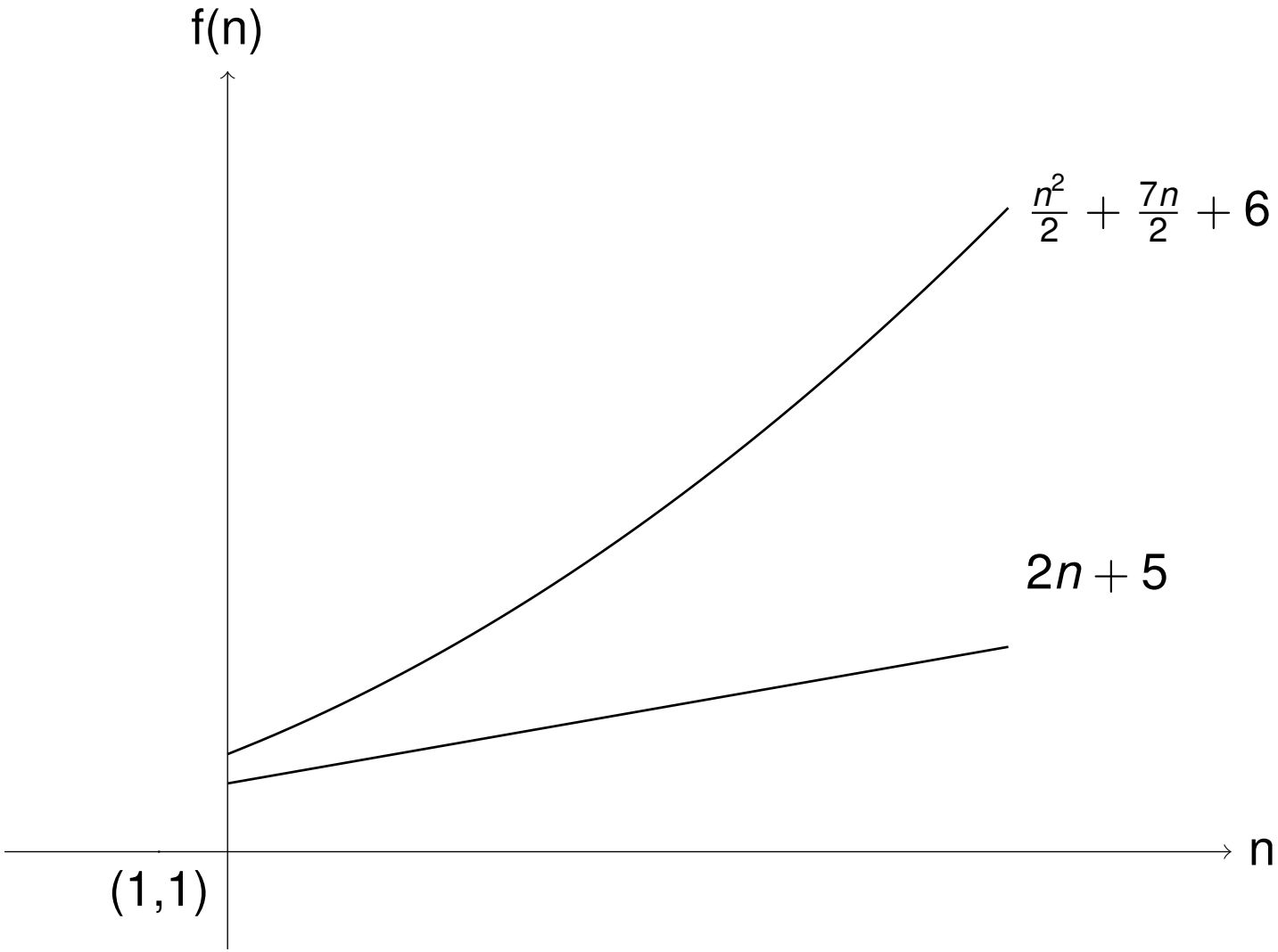
Equações



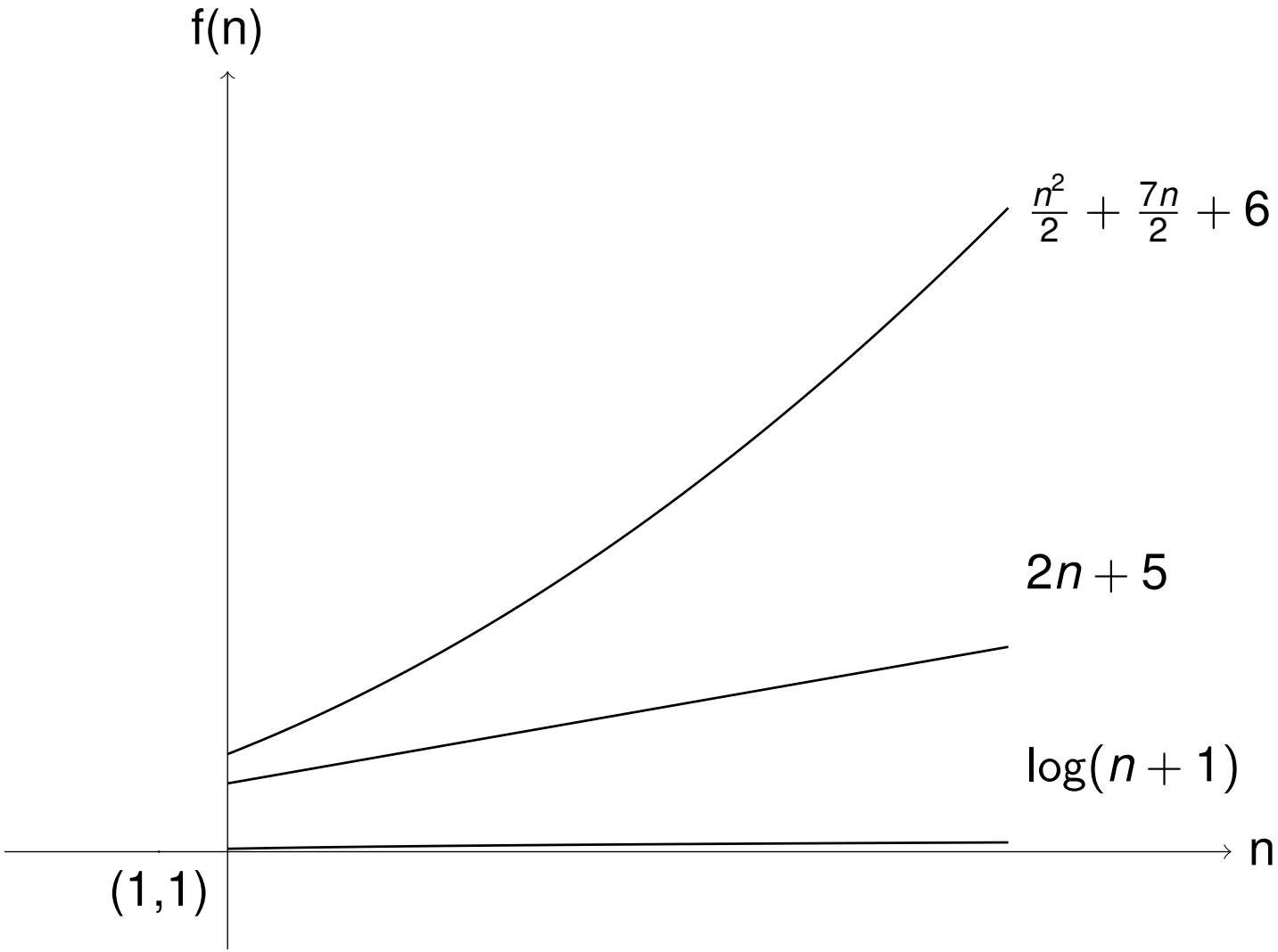
Equações



Equações



Equações



Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n			
$n\ log(n)$			
n^2			
n^3			
2^n			
3^n			

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$			
n^2			
n^3			
2^n			
3^n			

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$	$900\mu s$	$2.1ms$	$3.5ms$
n^2			
n^3			
2^n			
3^n			

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$	$900\mu s$	$2.1ms$	$3.5ms$
n^2	$4ms$	$16ms$	$36ms$
n^3			
2^n			
3^n			

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$	$900\mu s$	$2.1ms$	$3.5ms$
n^2	$4ms$	$16ms$	$36ms$
n^3	$80ms$	$640ms$	$2.16s$
2^n			
3^n			

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$	$900\mu s$	$2.1ms$	$3.5ms$
n^2	$4ms$	$16ms$	$36ms$
n^3	$80ms$	$640ms$	$2.16s$
2^n	$10s$	$27dias$	
3^n			

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$	$900\mu s$	$2.1ms$	$3.5ms$
n^2	$4ms$	$16ms$	$36ms$
n^3	$80ms$	$640ms$	$2.16s$
2^n	$10s$	$27dias$	3660 séculos
3^n			

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$	$900\mu s$	$2.1ms$	$3.5ms$
n^2	$4ms$	$16ms$	$36ms$
n^3	$80ms$	$640ms$	$2.16s$
2^n	$10s$	$27dias$	3660 séculos
3^n	$580min$		

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$	$900\mu s$	$2.1ms$	$3.5ms$
n^2	$4ms$	$16ms$	$36ms$
n^3	$80ms$	$640ms$	$2.16s$
2^n	$10s$	$27dias$	3660 séculos
3^n	$580min$	38550 séculos	

[1] Dura/custa: Termo de comparação dos algoritmos

Comportamento assintótico

Comparação numérica da complexidade: Considere que cada operação ($n = 1$) dura/custa[1] de $0.0001s = 100\mu s$.

$T(n)$	20	40	60
n	$200\mu s$	$400\mu s$	$600\mu s$
$n\ log(n)$	$900\mu s$	$2.1ms$	$3.5ms$
n^2	$4ms$	$16ms$	$36ms$
n^3	$80ms$	$640ms$	$2.16s$
2^n	$10s$	$27dias$	3660 séculos
3^n	$580min$	38550 séculos	$1.3 * 10^{14}$ séculos[2]

[1] Dura/custa: Termo de comparação dos algoritmos

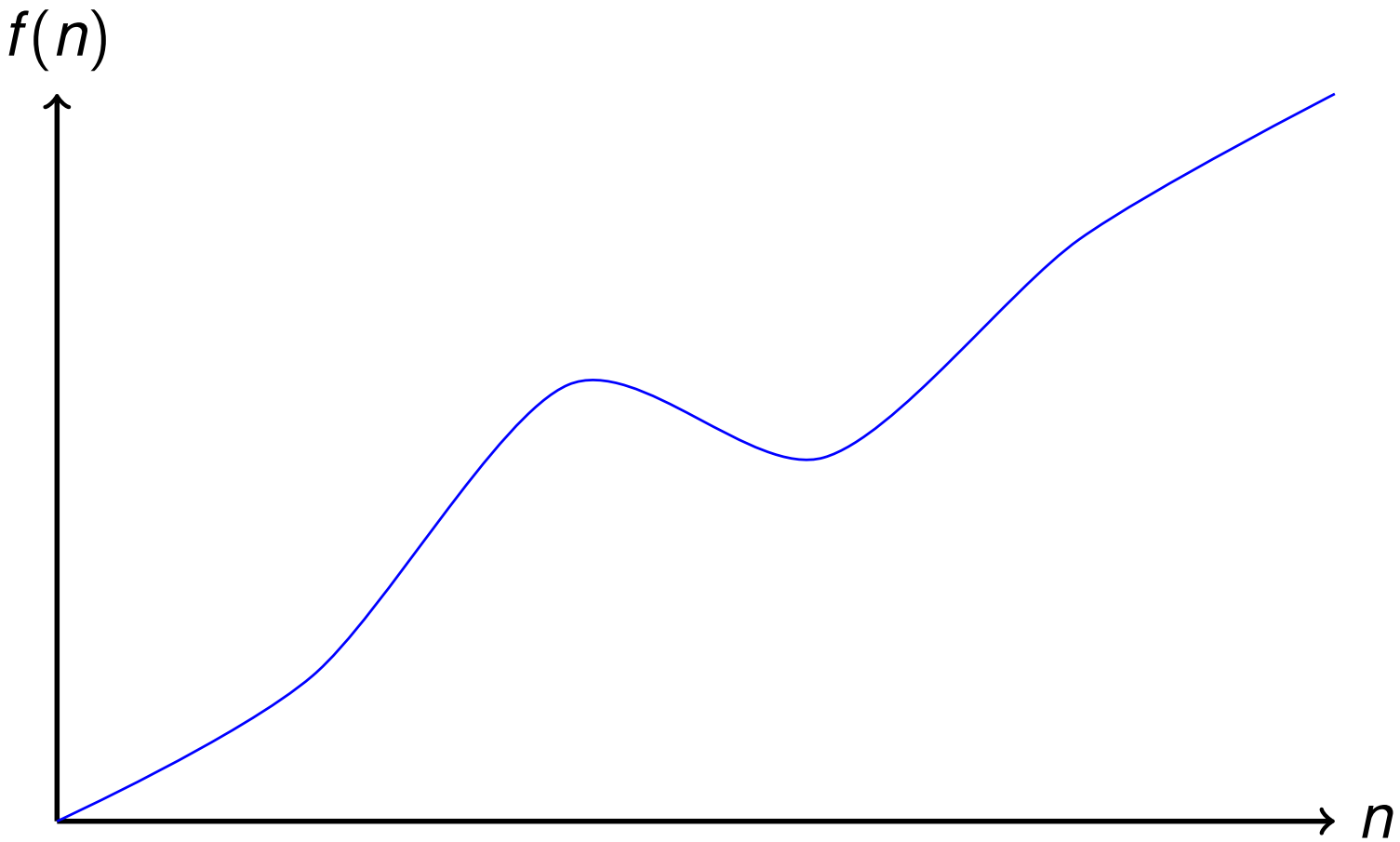
[2] O tempo do universo medido em séculos é de aproximadamente $13.8 * 10^7$ séculos

Complexidade Assintótica

O que significa complexidade assintótica?

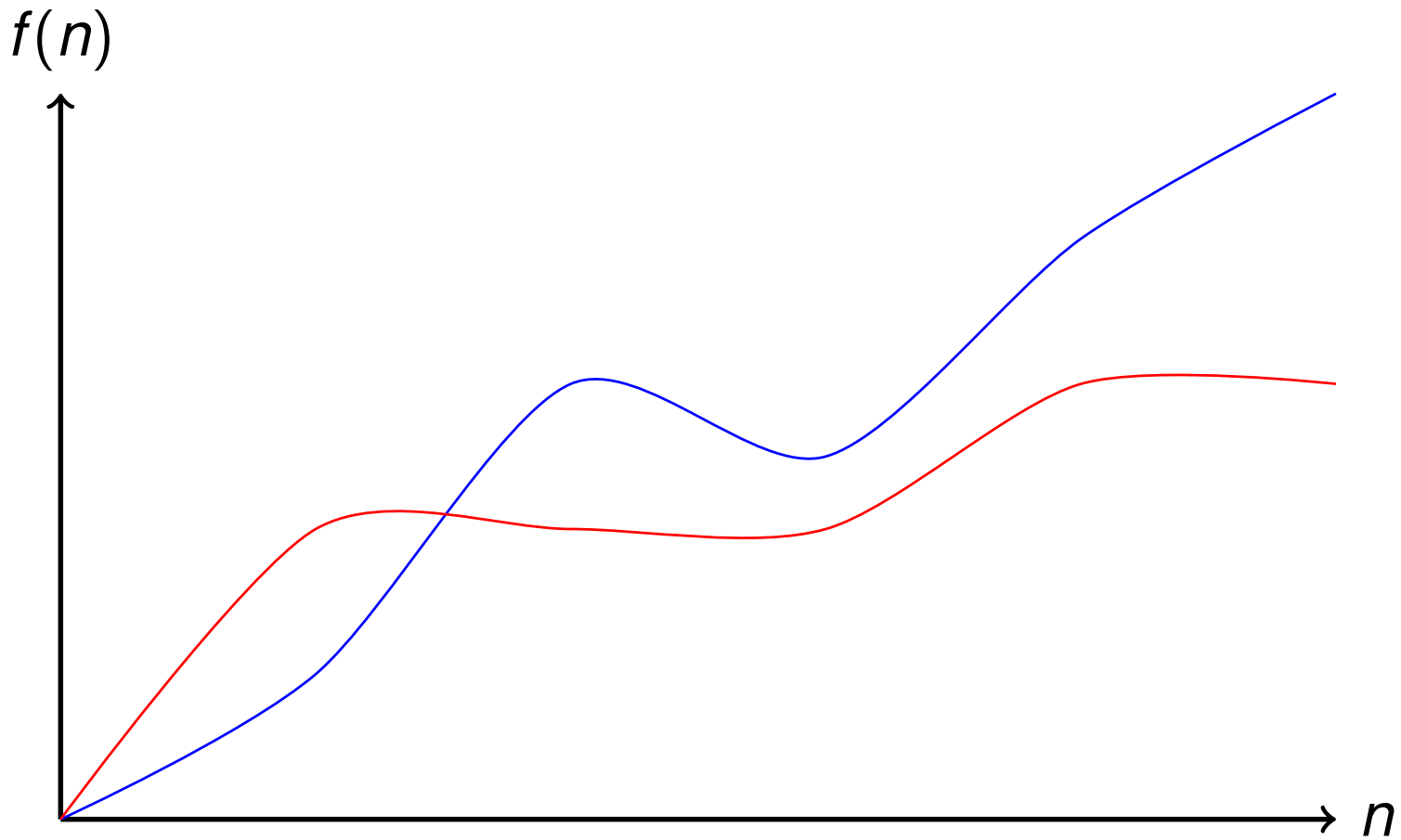
Complexidade Assintótica

O que significa complexidade assintótica?



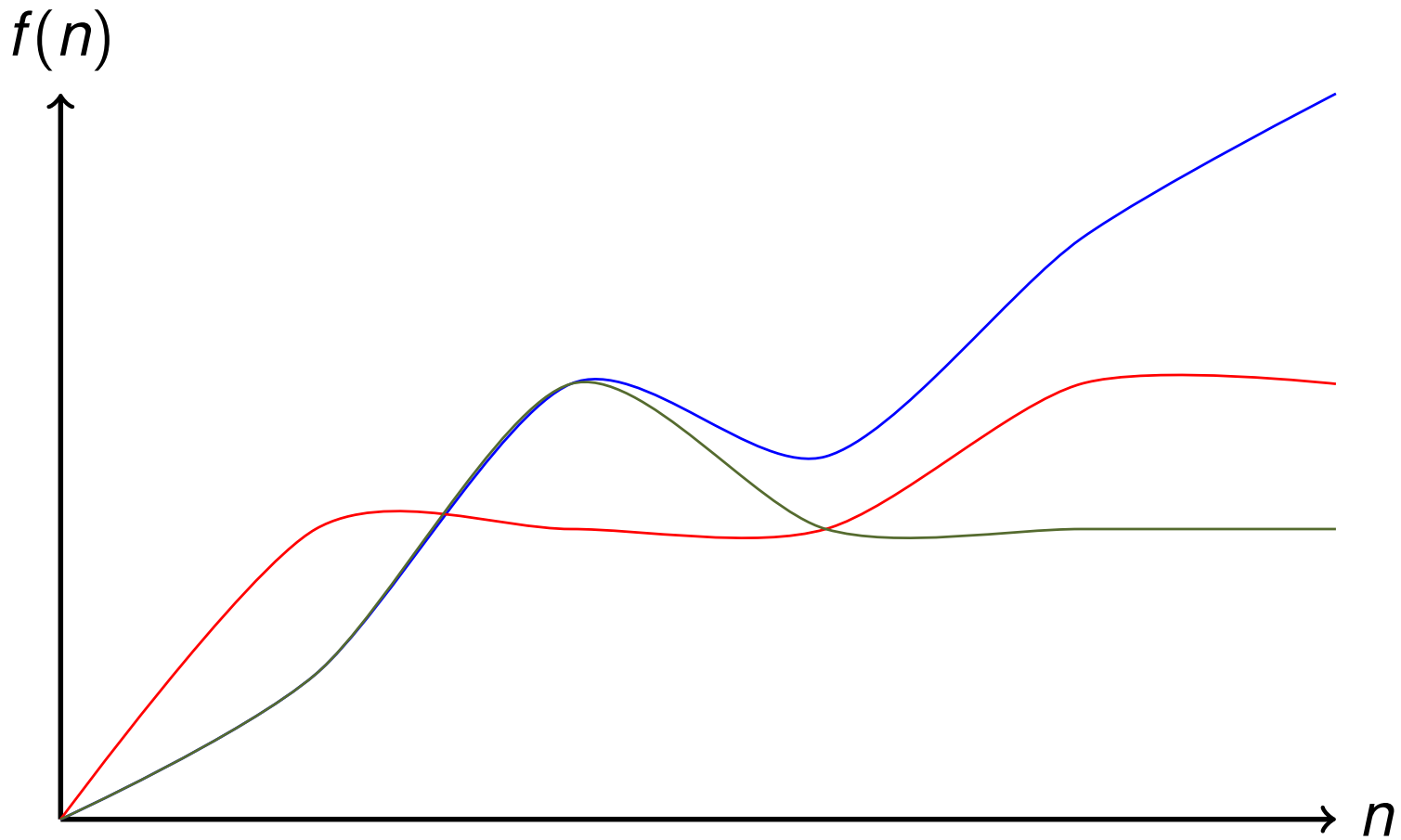
Complexidade Assintótica

O que significa complexidade assintótica?



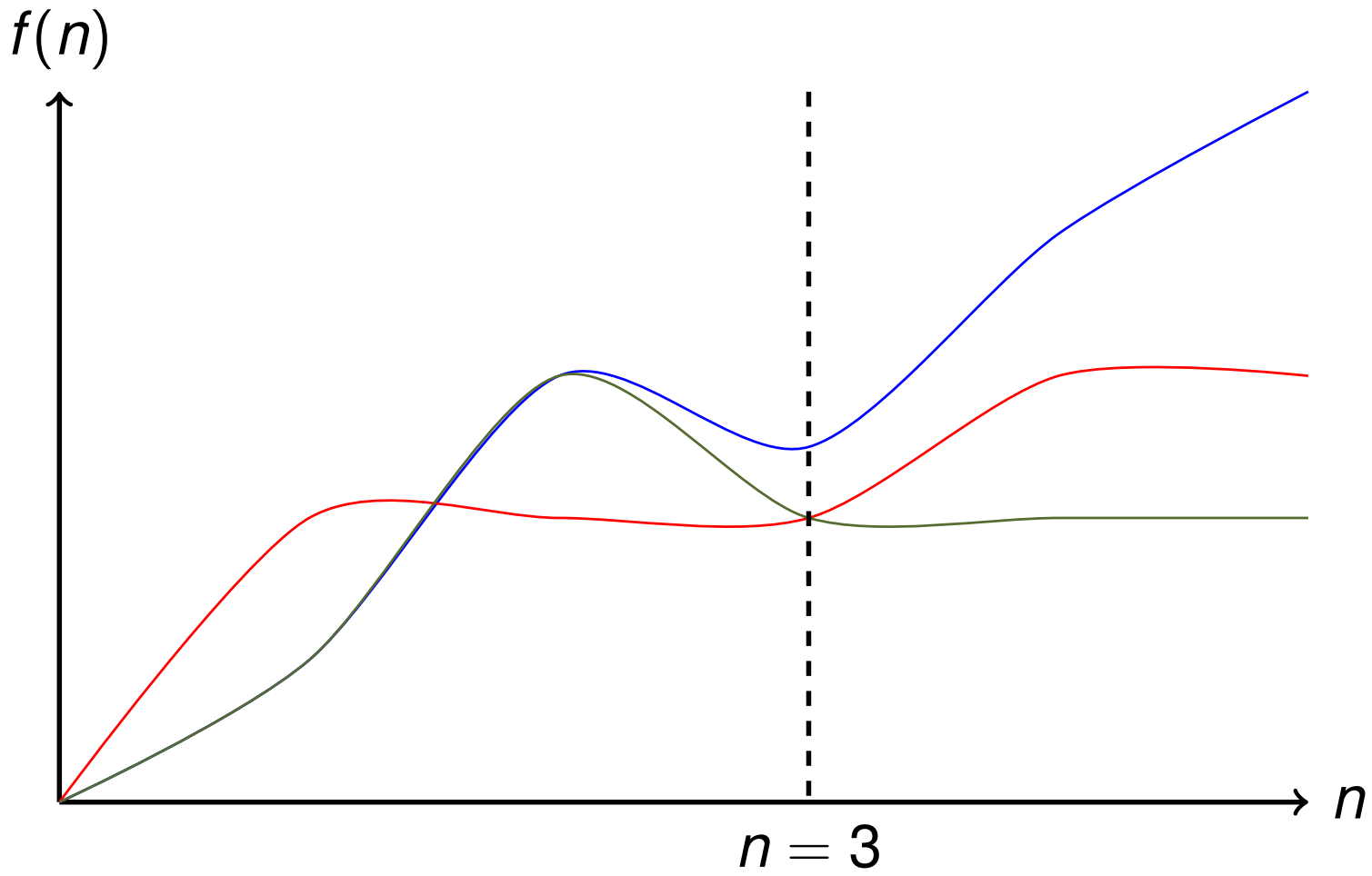
Complexidade Assintótica

O que significa complexidade assintótica?



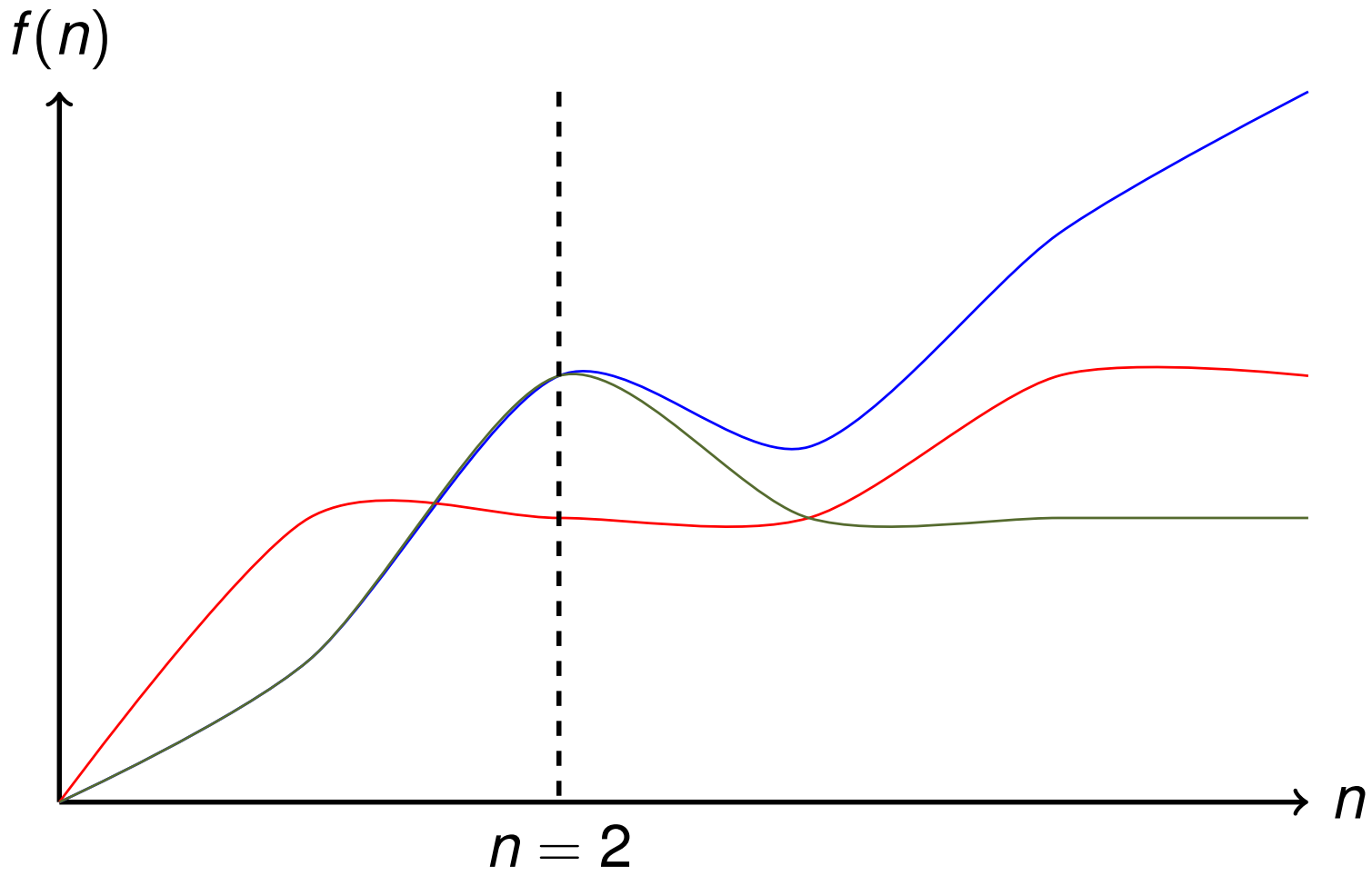
Complexidade Assintótica

O que significa complexidade assintótica?



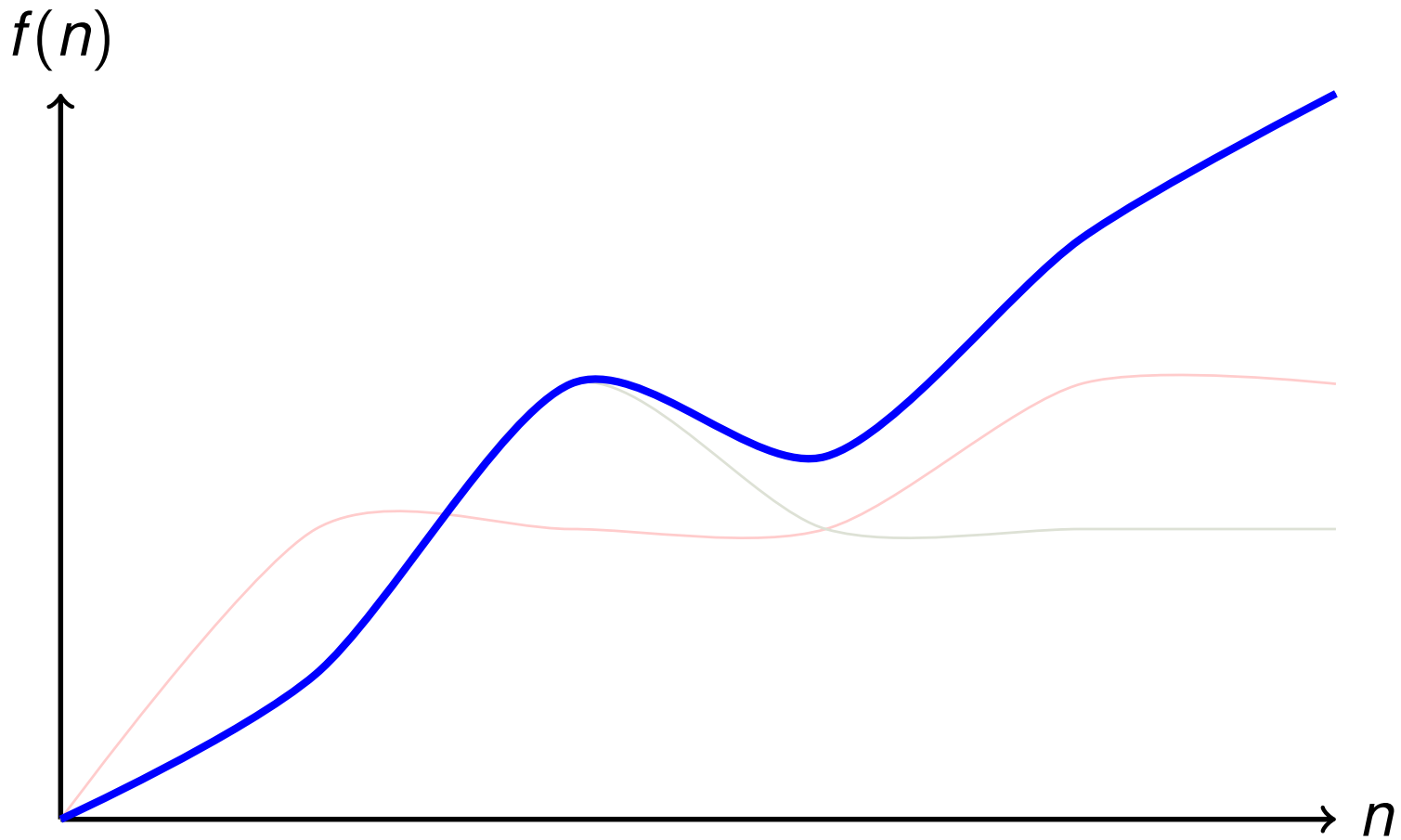
Complexidade Assintótica

O que significa complexidade assintótica?



Complexidade Assintótica

O que significa complexidade assintótica?



Assíntota

O que é uma assíntota?

São retas horizontais ou verticais que determinam a aproximação das curvas.

Exemplo

Assíntota

O que é uma assíntota?

São retas horizontais ou verticais que determinam a aproximação das curvas.

Exemplo

a) $f(x) = \frac{3x}{x-1}$

Assíntota

O que é uma assíntota?

São retas horizontais ou verticais que determinam a aproximação das curvas.

Exemplo

a) $f(x) = \frac{3x}{x-1}$

b) $f(x) = \frac{1}{(x-2)^2}$

Assíntota

O que é uma assíntota?

São retas horizontais ou verticais que determinam a aproximação das curvas.

Exemplo

a) $f(x) = \frac{3x}{x-1}$

b) $f(x) = \frac{1}{(x-2)^2}$

c) $f(x) = \frac{(x+2)(x-1)}{x(x+1)(x-2)}$

Assíntota

O que é uma assíntota?

São retas horizontais ou verticais que determinam a aproximação das curvas.

Exemplo

a) $f(x) = \frac{3x}{x-1}$

b) $f(x) = \frac{1}{(x-2)^2}$

c) $f(x) = \frac{(x+2)(x-1)}{x(x+1)(x-2)}$

d) $f(x) = \frac{x}{\sqrt{x^2+2}}$

Assíntota

O que é uma assíntota?

São retas horizontais ou verticais que determinam a aproximação das curvas.

Exemplo

a) $f(x) = \frac{3x}{x-1}$

b) $f(x) = \frac{1}{(x-2)^2}$

c) $f(x) = \frac{(x+2)(x-1)}{x(x+1)(x-2)}$

d) $f(x) = \frac{x}{\sqrt{x^2+2}}$

e) $f(x) = 1 + e^{-x}$

Assíntota

O que é uma assíntota?

São retas horizontais ou verticais que determinam a aproximação das curvas.

Exemplo

a) $f(x) = \frac{3x}{x-1}$

b) $f(x) = \frac{1}{(x-2)^2}$

c) $f(x) = \frac{(x+2)(x-1)}{x(x+1)(x-2)}$

d) $f(x) = \frac{x}{\sqrt{x^2+2}}$

e) $f(x) = 1 + e^{-x}$

f) $f(x) = \log(2 + 3^x)$

Complexidade Assintótica

Definição

Uma função $g(n)$ **domina assintoticamente** outra função f(n) se existem duas constantes positivas c e m tais que, para $n \geq m$, tem-se

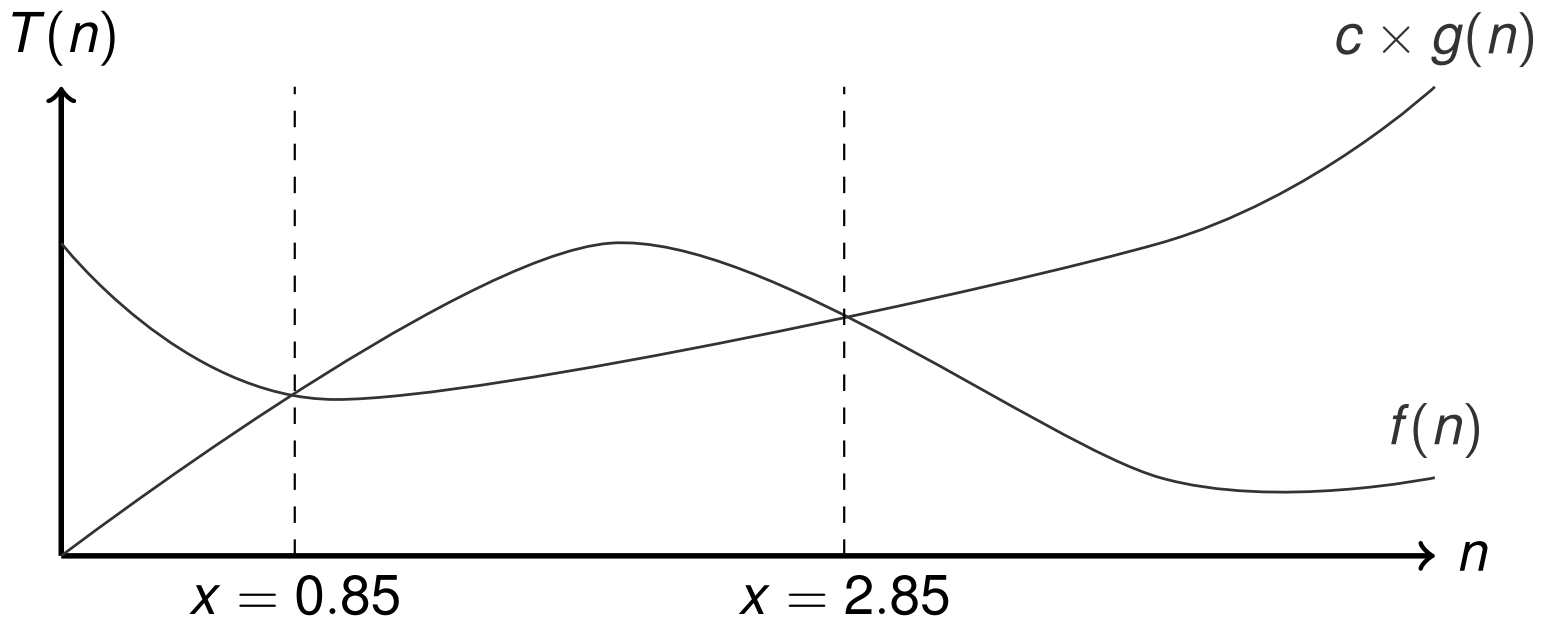
$$|f(n)| \leq c \times |g(n)|$$

Complexidade Assintótica

Definição

Uma função $g(n)$ **domina assintoticamente** outra função f(n) se existem duas constantes positivas c e m tais que, para $n \geq m$, tem-se

$$|f(n)| \leq c \times |g(n)|$$



Assíntota

Domínio assintótico

Determine quais funções são dominantes:

Assíntota

Domínio assintótico

Determine quais funções são dominantes:

a) $f(n) = n$ ou $g(n) = -n^2$

Assíntota

Domínio assintótico

Determine quais funções são dominantes:

a) $f(n) = n$ ou $g(n) = -n^2$

b) $f(n) = 50n$ ou $g(n) = 2n^2$

Assíntota

Domínio assintótico

Determine quais funções são dominantes:

a) $f(n) = n$ ou $g(n) = -n^2$

b) $f(n) = 50n$ ou $g(n) = 2n^2$

c) $f(n) = (n+1)^2$ ou $g(n) = n^2$

Assíntota

Domínio assintótico

Determine quais funções são dominantes:

a) $f(n) = n$ ou $g(n) = -n^2$

b) $f(n) = 50n$ ou $g(n) = 2n^2$

c) $f(n) = (n+1)^2$ ou $g(n) = n^2$

d) $f(n) = n^2 + 2n + 1$ ou $g(n) = 0.1n^3$

Assíntota

Domínio assintótico

Determine quais funções são dominantes:

a) $f(n) = n$ ou $g(n) = -n^2$

b) $f(n) = 50n$ ou $g(n) = 2n^2$

c) $f(n) = (n+1)^2$ ou $g(n) = n^2$

d) $f(n) = n^2 + 2n + 1$ ou $g(n) = 0.1n^3$

e) $f(n) = 0.2n^2$ ou $g(n) = 1000n$