

Análise de Algoritmos

CLRS 3.1 e 3.2
AU 3.3, 3.4 e 3.6

Essas transparências foram adaptadas das transparências do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Notação O

Intuitivamente...

$O(f(n)) \approx$ funções que não crescem mais rápido que $f(n)$
 \approx funções menores ou iguais a um múltiplo de $f(n)$

n^2 $(3/2)n^2$ $9999n^2$ $n^2/1000$ etc.

crescem todas com a **mesma velocidade**

Notação O

Intuitivamente...

$O(f(n)) \approx$ funções que não crescem mais rápido que $f(n)$
 \approx funções menores ou iguais a um múltiplo de $f(n)$

n^2 $(3/2)n^2$ $9999n^2$ $n^2/1000$ etc.

crescem todas com a **mesma velocidade**

● $n^2 + 99n$ é $O(n^2)$

● $33n^2$ é $O(n^2)$

● $9n + 2$ é $O(n^2)$

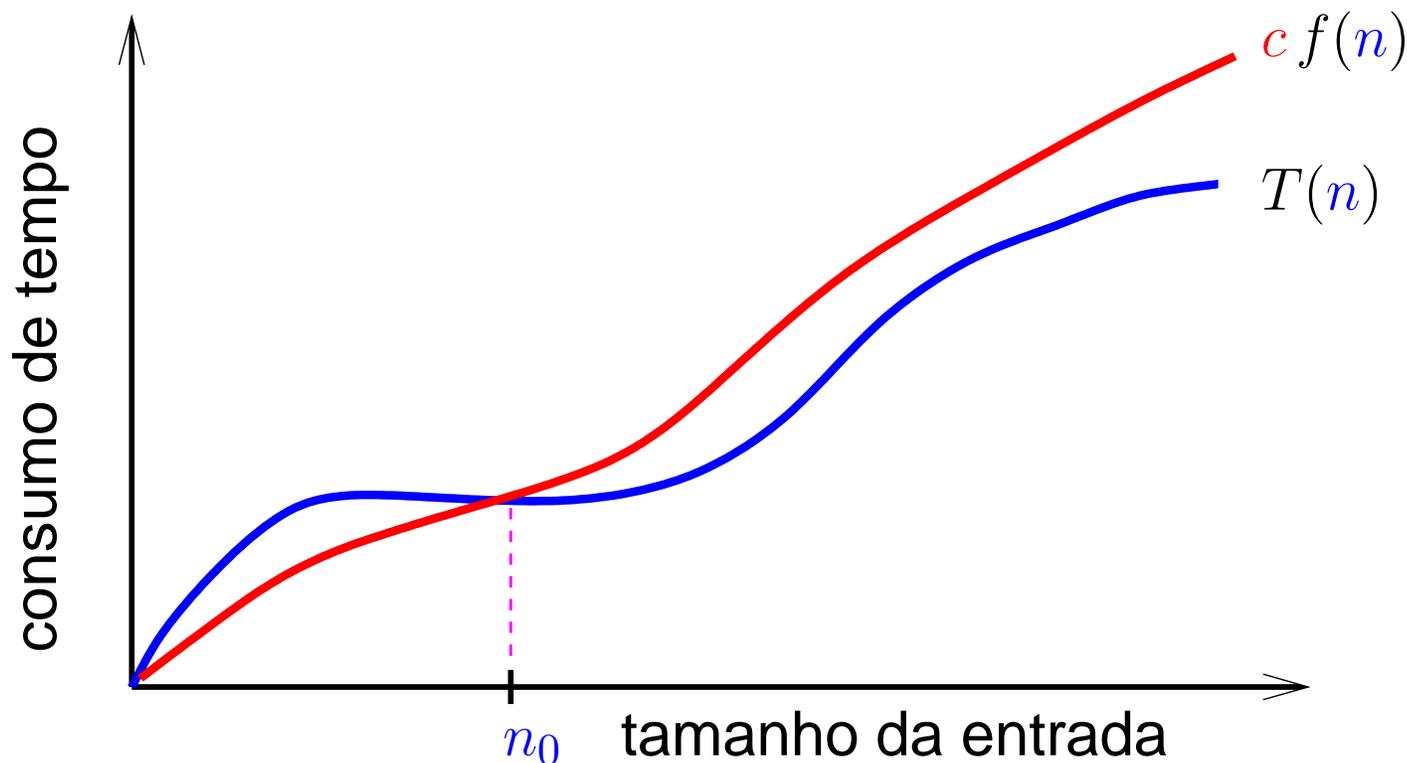
● $0,00001n^3 - 200n^2$ **não é** $O(n^2)$

Definição

Sejam $T(n)$ e $f(n)$ funções dos inteiros nos reais.
Dizemos que $T(n)$ é $O(f(n))$ se existem constantes positivas c e n_0 tais que

$$T(n) \leq c f(n)$$

para todo $n \geq n_0$.

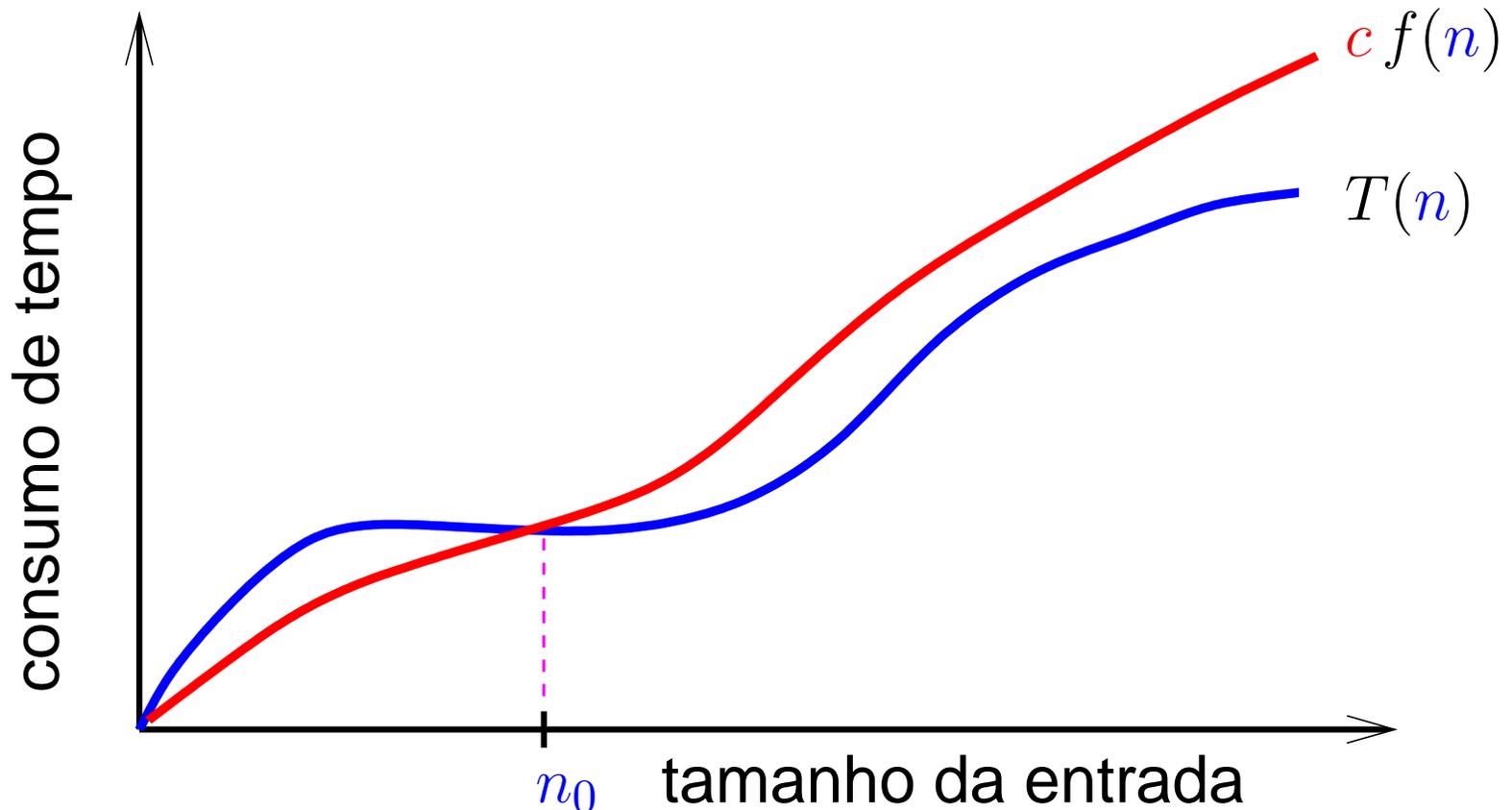


Mais informal

$T(n)$ é $O(f(n))$ se existe $c > 0$ tal que

$$T(n) \leq c f(n)$$

para todo n suficientemente **GRANDE**.



Ordenação por inserção

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$        $i \leftarrow j - 1$ 
3      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
4           $A[i + 1] \leftarrow A[i]$   ▷ desloca
5           $i \leftarrow i - 1$ 
6       $A[i + 1] \leftarrow chave$   ▷ insere
```

Ordenação por inserção

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$        $i \leftarrow j - 1$ 
3      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
4           $A[i + 1] \leftarrow A[i]$   ▷ desloca
5           $i \leftarrow i - 1$ 
6       $A[i + 1] \leftarrow chave$   ▷ insere
```

linha consumo de todas as execuções da linha

| | |
|---|---|
| 1 | ? |
| 2 | ? |
| 3 | ? |
| 4 | ? |
| 5 | ? |
| 6 | ? |

total ?

Ordenação por inserção

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$        $i \leftarrow j - 1$ 
3      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
4           $A[i + 1] \leftarrow A[i]$   ▷ desloca
5           $i \leftarrow i - 1$ 
6       $A[i + 1] \leftarrow chave$   ▷ insere
```

linha consumo de todas as execuções da linha

1 $O(n)$

2 $O(n)$

3 $O(n^2)$

4 $O(n^2)$

5 $O(n^2)$

6 $O(n)$

total $O(3n^2 + 3n) = O(n^2)$

Conclusão

O algoritmo ORDENA-POR-INSERÇÃO consome $O(n^2)$ unidades de tempo.

Também escreve-se

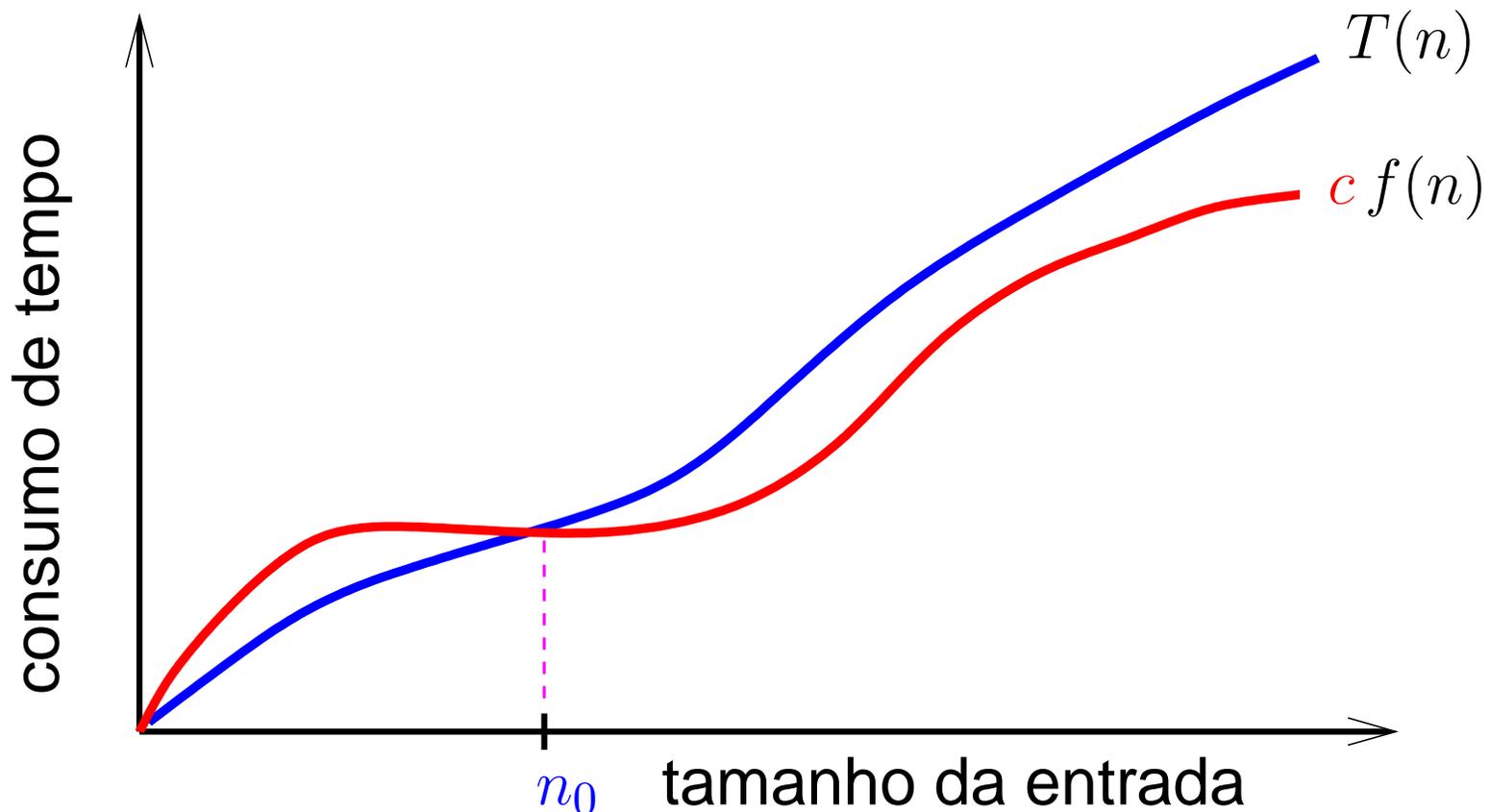
O algoritmo ORDENA-POR-INSERÇÃO consome tempo $O(n^2)$.

Notação Omega

Dizemos que $T(n)$ é $\Omega(f(n))$ se existem constantes positivas c e n_0 tais que

$$c f(n) \leq T(n)$$

para todo $n \geq n_0$.

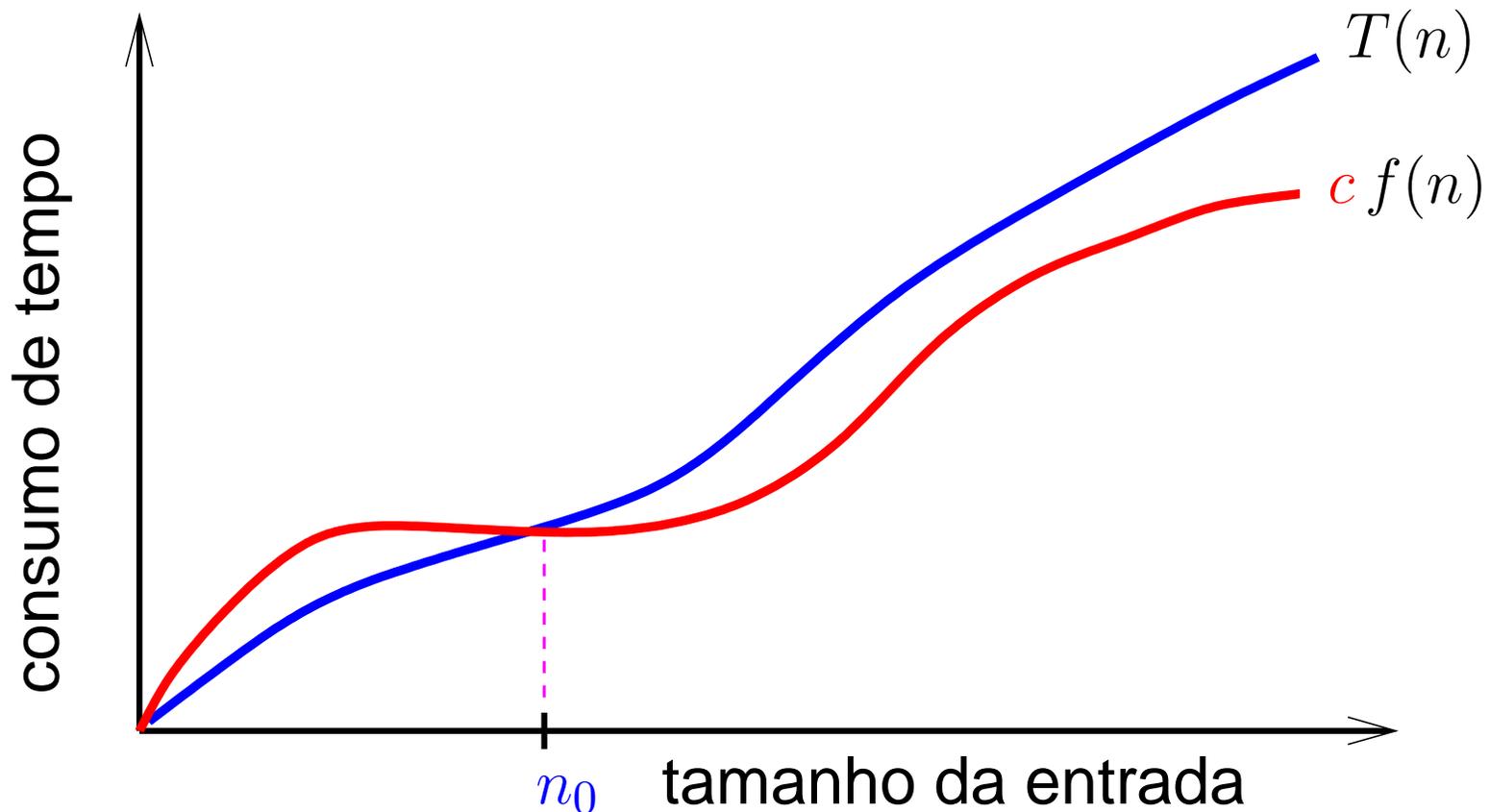


Mais informal

$T(n) = \Omega(f(n))$ se existe $c > 0$ tal que

$$c f(n) \leq T(n)$$

para todo n **suficientemente GRANDE.**



Exemplos

Exemplo 1

Se $T(n) \geq 0.001n^2$ para todo $n \geq 8$, então $T(n)$ é $\Omega(n^2)$.

Exemplos

Exemplo 1

Se $T(n) \geq 0.001n^2$ para todo $n \geq 8$, então $T(n)$ é $\Omega(n^2)$.

Prova: Aplique a definição com $c = 0.001$ e $n_0 = 8$.

Exemplo 2

O consumo de tempo do **ORDENA-POR-INSERÇÃO** é $O(n^2)$ e é $\Omega(n)$.

Exemplo 2

O consumo de tempo do **ORDENA-POR-INSERÇÃO** é $O(n^2)$ e é $\Omega(n)$.

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça  
2       $chave \leftarrow A[j]$   
3       $i \leftarrow j - 1$   
4      enquanto  $i \geq 1$  e  $A[i] > chave$  faça  
5           $A[i + 1] \leftarrow A[i]$     ▷ desloca  
6           $i \leftarrow i - 1$   
7       $A[i + 1] \leftarrow chave$     ▷ insere
```

Exemplo 2

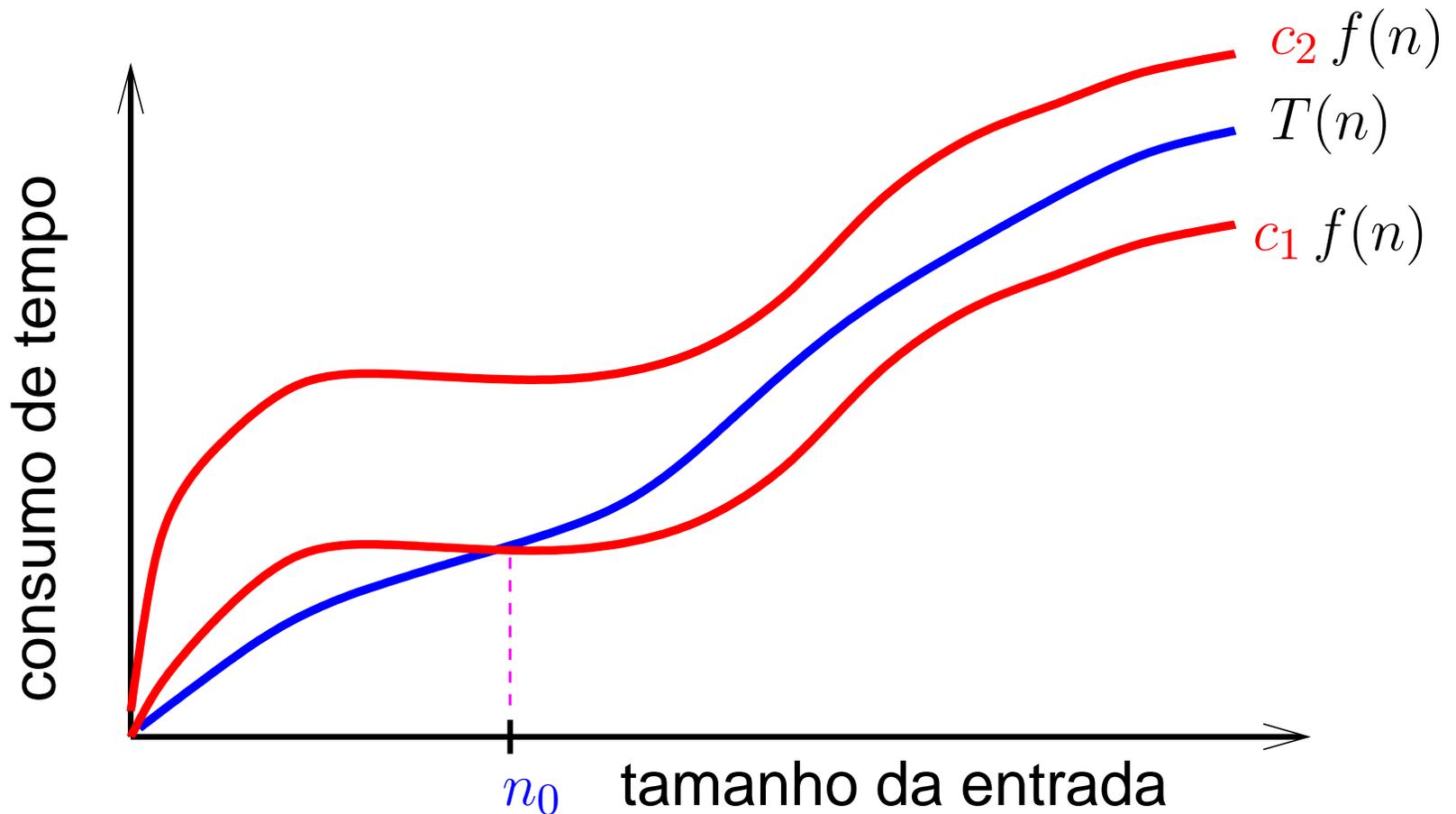
O consumo de tempo do **ORDENA-POR-INSERÇÃO** é $O(n^2)$ e é $\Omega(n)$.

| linha | todas as execuções da linha |
|--------------|---|
| 1 | = n |
| 2-3 | = $n - 1$ |
| 4 | $\geq n - 1$ |
| 5-6 | ≥ 0 |
| 7 | = $n - 1$ |
| total | $\geq 4n - 3 = \Omega(n)$ |

Notação Theta

Sejam $T(n)$ e $f(n)$ funções dos inteiros no reais.
Dizemos que $T(n)$ é $\Theta(f(n))$ se

$T(n)$ é $O(f(n))$ e $T(n)$ é $\Omega(f(n))$.

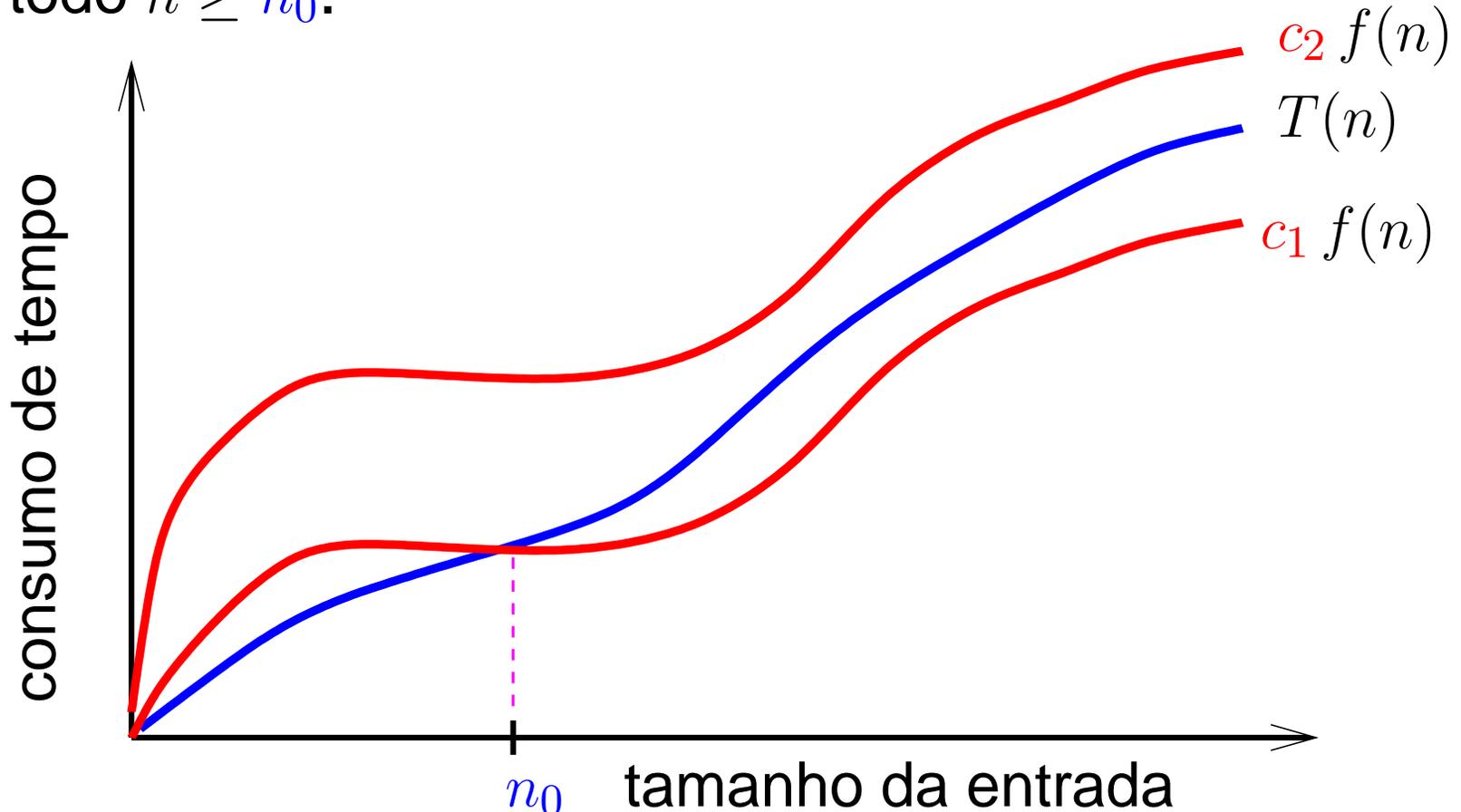


Notação Theta

Dizemos que $T(n)$ é $\Theta(f(n))$ se se existem constantes positivas c_1, c_2 e n_0 tais que

$$c_1 f(n) \leq T(n) \leq c_2 f(n)$$

para todo $n \geq n_0$.



Intuitivamente

Comparação **assintótica**, ou seja, para n **ENORME**.

| comparação | comparação assintótica |
|------------------|-------------------------|
| $T(n) \leq f(n)$ | $T(n)$ é $O(f(n))$ |
| $T(n) \geq f(n)$ | $T(n)$ é $\Omega(f(n))$ |
| $T(n) = f(n)$ | $T(n)$ é $\Theta(f(n))$ |

Tamanho máximo de problemas

Suponha que cada operação consome 1 microsegundo ($1\mu s$).

| consumo de tempo (μs) | Tamanho máximo de problemas (n) | | |
|------------------------------|-------------------------------------|----------|---------|
| | 1 segundo | 1 minuto | 1 hora |
| $400n$ | 2500 | 150000 | 9000000 |
| $20n \lceil \lg n \rceil$ | 4096 | 166666 | 7826087 |
| $2n^2$ | 707 | 5477 | 42426 |
| n^4 | 31 | 88 | 244 |
| 2^n | 19 | 25 | 31 |

Michael T. Goodrich e Roberto Tamassia, *Projeto de Algoritmos*, Bookman.

Crescimento de algumas funções

| n | $\lg n$ | \sqrt{n} | $n \lg n$ | n^2 | n^3 | 2^n |
|------|---------|------------|-----------|---------|------------------|----------------------|
| 2 | 1 | 1,4 | 2 | 4 | 8 | 4 |
| 4 | 2 | 2 | 8 | 16 | 64 | 16 |
| 8 | 3 | 2,8 | 24 | 64 | 512 | 256 |
| 16 | 4 | 4 | 64 | 256 | 4096 | 65536 |
| 32 | 5 | 5,7 | 160 | 1024 | 32768 | 4294967296 |
| 64 | 6 | 8 | 384 | 4096 | 262144 | $1,8 \cdot 10^{19}$ |
| 128 | 7 | 11 | 896 | 16384 | 2097152 | $3,4 \cdot 10^{38}$ |
| 256 | 8 | 16 | 1048 | 65536 | 16777216 | $1,1 \cdot 10^{77}$ |
| 512 | 9 | 23 | 4608 | 262144 | 134217728 | $1,3 \cdot 10^{154}$ |
| 1024 | 10 | 32 | 10240 | 1048576 | $1,1 \cdot 10^9$ | $1,7 \cdot 10^{308}$ |

Nomes de classes Θ

| classe | nome |
|------------------------------|-------------|
| $\Theta(1)$ | constante |
| $\Theta(\log n)$ | logarítmica |
| $\Theta(n)$ | linear |
| $\Theta(n \log n)$ | $n \log n$ |
| $\Theta(n^2)$ | quadrática |
| $\Theta(n^3)$ | cúbica |
| $\Theta(n^k)$ com $k \geq 1$ | polinomial |
| $\Theta(2^n)$ | exponencial |
| $\Theta(a^n)$ com $a > 1$ | exponencial |

Palavras de Cautela

Suponha que A e B são algoritmos para um mesmo problema. Suponha que o consumo de tempo de A é “essencialmente” $100n$ e que o consumo de tempo de B é “essencialmente” $n \log_{10} n$.

Palavras de Cautela

Suponha que A e B são algoritmos para um mesmo problema. Suponha que o consumo de tempo de A é “essencialmente” $100n$ e que o consumo de tempo de B é “essencialmente” $n \log_{10} n$.

$100n$ é $\Theta(n)$ e $n \log_{10} n$ é $\Theta(n \lg n)$.

Logo, A é **assintoticamente** mais eficiente que B .

Palavras de Cautela

Suponha que A e B são algoritmos para um mesmo problema. Suponha que o consumo de tempo de A é “essencialmente” $100n$ e que o consumo de tempo de B é “essencialmente” $n \log_{10} n$.

$100n$ é $\Theta(n)$ e $n \log_{10} n$ é $\Theta(n \lg n)$.

Logo, A é **assintoticamente** mais eficiente que B .

A é mais eficiente que B para $n \geq 10^{100}$.

10^{100} = um *googol*

\approx número de átomos no universo observável

= número **ENORME**

Palavras de Cautela

Conclusão:

Lembre das constantes e termos de baixa ordem que estão “**escondidos**” na notação assintótica.

Em geral um algoritmo que consome tempo $\Theta(n \lg n)$, e com fatores constantes razoáveis, é bem eficiente.

Um algoritmo que consome tempo $\Theta(n^2)$ pode, algumas vezes, ser satisfatório.

Um algoritmo que consome tempo $\Theta(2^n)$ é dificilmente aceitável.

Do ponto de vista de **AA**, **eficiente = polinomial**.

Análise do caso médio

Consideremos que a entrada do algoritmo é uma permutação de 1 a n escolhida com probabilidade uniforme.

Análise do caso médio

Consideremos que a entrada do algoritmo é uma permutação de 1 a n escolhida com probabilidade uniforme.

Qual é o número esperado de execuções da linha 4?

Análise do caso médio

Consideremos que a entrada do algoritmo é uma permutação de 1 a n escolhida com probabilidade uniforme.

Qual é o número esperado de execuções da linha 4?

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$ 
3       $i \leftarrow j - 1$ 
4      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
5           $A[i + 1] \leftarrow A[i]$     ▷ desloca
6           $i \leftarrow i - 1$ 
7       $A[i + 1] \leftarrow chave$     ▷ insere
```

Análise do caso médio

Consideremos que a entrada do algoritmo é uma permutação de 1 a n escolhida com probabilidade uniforme.

Qual é o número esperado de execuções da linha 4?

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$ 
3       $i \leftarrow j - 1$ 
4      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
5           $A[i + 1] \leftarrow A[i]$     ▷ desloca
6           $i \leftarrow i - 1$ 
7       $A[i + 1] \leftarrow chave$     ▷ insere
```

Análise do caso médio

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$ 
3       $i \leftarrow j - 1$ 
4      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
5           $A[i + 1] \leftarrow A[i]$     ▷ desloca
6           $i \leftarrow i - 1$ 
7       $A[i + 1] \leftarrow chave$     ▷ insere
```

Para cada valor de j , a linha 4 é executada de 1 a j vezes.

Análise do caso médio

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$ 
3       $i \leftarrow j - 1$ 
4      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
5           $A[i + 1] \leftarrow A[i]$     ▷ desloca
6           $i \leftarrow i - 1$ 
7       $A[i + 1] \leftarrow chave$     ▷ insere
```

Para cada valor de j , a linha 4 é executada de 1 a j vezes.

Qual é a probabilidade dela ser executada t vezes?

Análise do caso médio

ORDENA-POR-INSERÇÃO (A, n)

```
1  para  $j \leftarrow 2$  até  $n$  faça
2       $chave \leftarrow A[j]$ 
3       $i \leftarrow j - 1$ 
4      enquanto  $i \geq 1$  e  $A[i] > chave$  faça
5           $A[i + 1] \leftarrow A[i]$     ▷ desloca
6           $i \leftarrow i - 1$ 
7       $A[i + 1] \leftarrow chave$     ▷ insere
```

Para cada valor de j , a linha 4 é executada de 1 a j vezes.

Qual é a probabilidade dela ser executada t vezes?

Esta probabilidade é $1/j$. (Explicação na aula.)

Análise do caso médio

Para cada valor de j , a linha 4 é executada de 1 a j vezes.

Qual é a probabilidade dela ser executada t vezes?

Esta probabilidade é $1/j$.

Análise do caso médio

Para cada valor de j , a linha 4 é executada de 1 a j vezes.

Qual é a probabilidade dela ser executada t vezes?

Esta probabilidade é $1/j$.

Portanto o número esperado de execuções da linha 4 é

$$\sum_{t=1}^j t \frac{1}{j} = \frac{1}{j} \sum_{t=1}^j t = \frac{1}{j} \frac{(j+1)j}{2} = \frac{j+1}{2}.$$

Análise do caso médio

Para cada valor de j , a linha 4 é executada de 1 a j vezes.

Qual é a probabilidade dela ser executada t vezes?

Esta probabilidade é $1/j$.

Portanto o número esperado de execuções da linha 4 é

$$\sum_{t=1}^j t \frac{1}{j} = \frac{1}{j} \sum_{t=1}^j t = \frac{1}{j} \frac{(j+1)j}{2} = \frac{j+1}{2}.$$

E o número esperado de execuções da linha 4 no total é

$$\sum_{j=2}^n \frac{j+1}{2} = \frac{1}{2} \sum_{j=2}^n (j+1) = \frac{(n+4)(n-1)}{4} = \Theta(n^2).$$

Ordenação

$A[1..n]$ é **crecente** se $A[1] \leq \dots \leq A[n]$.

Problema: Rearranjar um vetor $A[1..n]$ de modo que ele fique crescente.

Ordenação

$A[1 \dots n]$ é **crecente** se $A[1] \leq \dots \leq A[n]$.

Problema: Rearranjar um vetor $A[1 \dots n]$ de modo que ele fique crescente.

ORDENA-POR-INSERÇÃO consome tempo $O(n^2)$ no pior caso e $\Theta(n^2)$ no caso médio.

Ordenação

$A[1..n]$ é **crecente** se $A[1] \leq \dots \leq A[n]$.

Problema: Rearranjar um vetor $A[1..n]$ de modo que ele fique crescente.

ORDENA-POR-INSERÇÃO consome tempo $O(n^2)$ no pior caso e $\Theta(n^2)$ no caso médio.

Conseguimos fazer melhor?

Ordenação

$A[1 \dots n]$ é **crecente** se $A[1] \leq \dots \leq A[n]$.

Problema: Rearranjar um vetor $A[1 \dots n]$ de modo que ele fique crescente.

ORDENA-POR-INSERÇÃO consome tempo $O(n^2)$ no pior caso e $\Theta(n^2)$ no caso médio.

Conseguimos fazer melhor?

Sim! Usando **divisão e conquista!**