Análise de Algoritmos

Parte destes slides são adaptações de slides

do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

18 de agosto de 2022

Análise de Algoritmos

CLRS 3.1 e 3.2

AU 3.3, 3.4 e 3.6

Notação O

Intuitivamente...

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

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

crescem todas com a mesma velocidade

Notação O

Intuitivamente...

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

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

crescem todas com a mesma velocidade

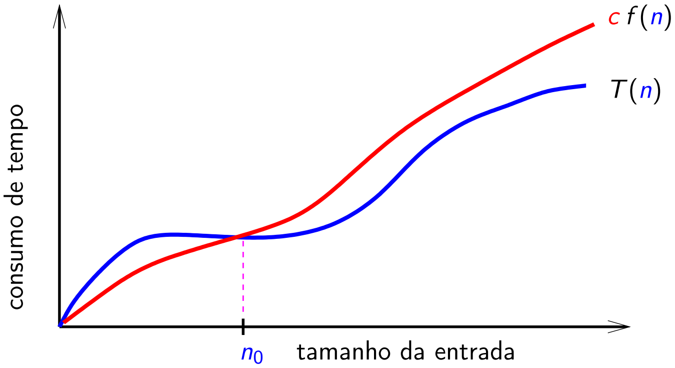
- $n^2 + 99n$ é $\mathrm{O}(n^2)$
- $33n^2$ é $\mathrm{O}(n^2)$
- $9n + 2$ é $\mathrm{O}(n^2)$
- $0{,}00001n^3 - 200n^2$ não é $\mathrm{O}(n^2)$

Definição

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

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

para todo $n \geq n_0$.

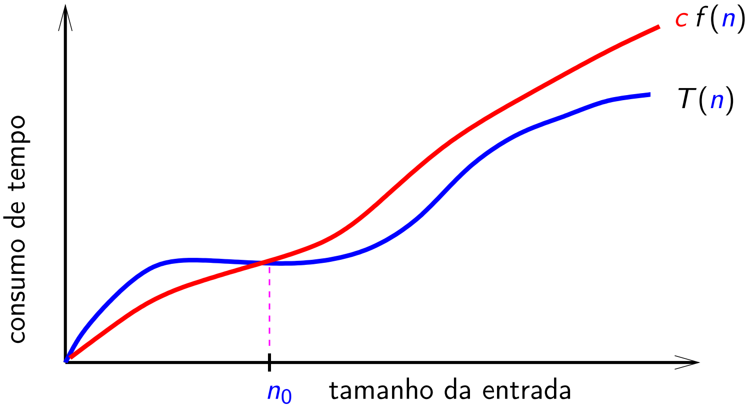


Mais informal

$T(n)$ é $\mathrm{O}(f(n))$ 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)

para $j \leftarrow 2$ **até** n **faça**
    $chave \leftarrow A[j]$ $i \leftarrow j - 1$
    enquanto $i \geq 1$ **e** $A[i] > chave$ **faça**
        $A[i+1] \leftarrow A[i]$ $\triangleright$ desloca
        $i \leftarrow i - 1$
    $A[i+1] \leftarrow chave$ $\triangleright$ insere

Ordenação por inserção

```
ORDENA-POR-INSERÇÃO (A, n)
1  para j ← 2 até n faça
2      chave ← A[j]        i ← j − 1
3      enquanto i ≥ 1 e A[i] > chave faça
4          A[i + 1] ← A[i]        ▷ desloca
5          i ← i − 1
6      A[i + 1] ← 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 ← 2 até n faça
2      chave ← A[j]      i ← j − 1
3      enquanto i ≥ 1 e A[i] > chave faça
4          A[i + 1] ← A[i]      ▷ desloca
5          i ← i − 1
6      A[i + 1] ← chave      ▷ insere
```

linha	consumo de todas as execuções da linha
1	$\mathrm{O}(n)$
2	$\mathrm{O}(n)$
3	$\mathrm{O}(n^2)$
4	$\mathrm{O}(n^2)$
5	$\mathrm{O}(n^2)$
6	$\mathrm{O}(n)$
total	$\mathrm{O}(3n^2 + 3n) = \mathrm{O}(n^2)$

Conclusão

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

delimitação superior

Também escreve-se

O algoritmo ORDENA-POR-INSERÇÃO consome
tempo $\mathrm{O}(n^2)$.

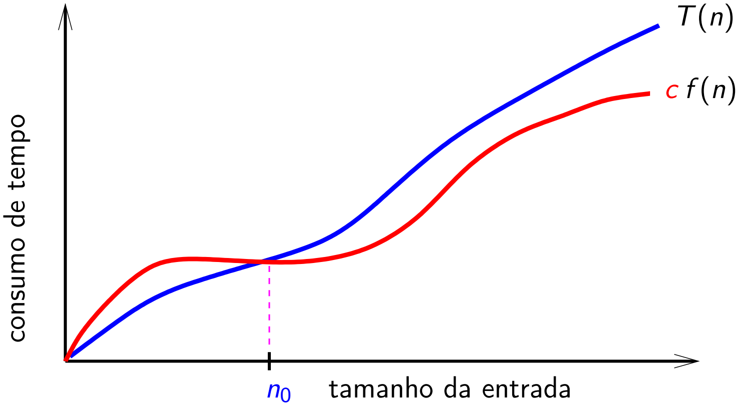
Notação Omega

delimitação inferior

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 $n \geq n_0$.

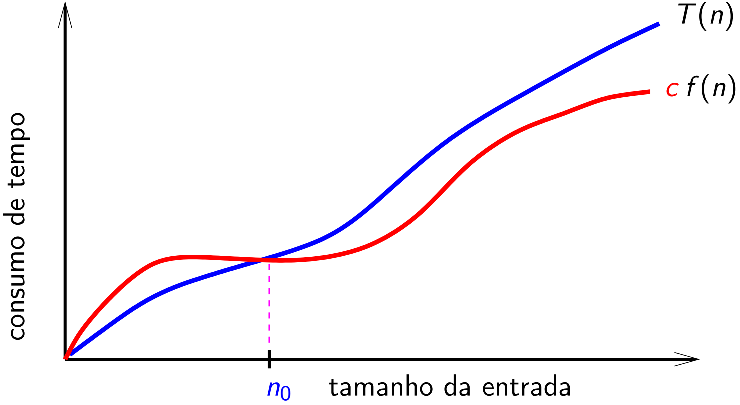


Mais informal

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

$$c\, f(n) \le T(n)$$

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 $\mathrm{O}(n^2)$ e $\Omega(n)$.

Exemplo 2

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

ORDENA-POR-INSERÇÃO (A, n)

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

Exemplo 2

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

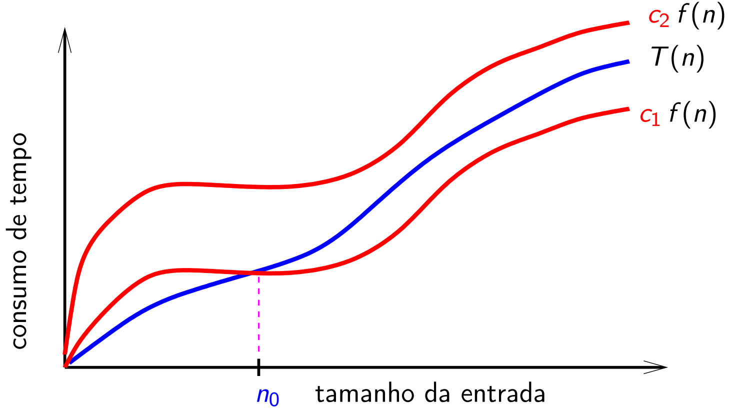
ORDENA-POR-INSERÇÃO (A, n)
1 **para** $j \leftarrow 2$ **até** n **faça**
2 $\quad$ $chave \leftarrow A[j]$
3 $\quad$ $i \leftarrow j - 1$
4 $\quad$ **enquanto** $i \geq 1$ **e** $A[i] > chave$ **faça**
5 $\quad\quad$ $A[i+1] \leftarrow A[i]$ $\quad \triangleright$ desloca
6 $\quad\quad$ $i \leftarrow i - 1$
7 $\quad$ $A[i+1] \leftarrow chave$ $\quad \triangleright$ insere

linha	todas as execuções da linha	
1	$=$	n
2-3	$=$	$n-1$
4	$\geq$	$n-1$
5-6	$\geq$	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   não oscila, fica na mesma ordem (Θ)

$T(n)$ é $\mathrm{O}(f(n))$ e $T(n)$ é $\Omega(f(n))$.   é $\Theta(n)$ quando é $O(n)$ e $\Omega(n)$

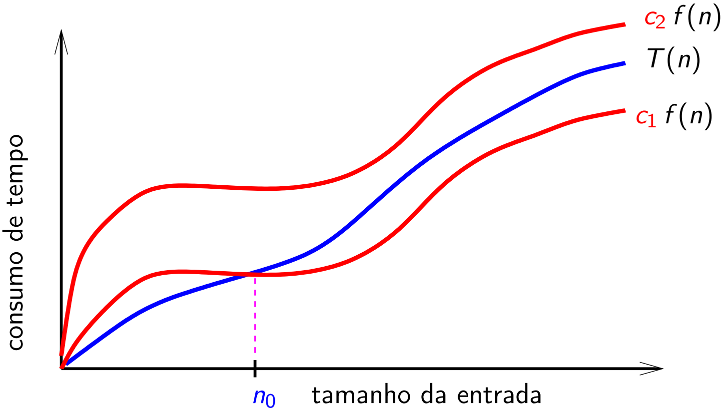


Notação Theta

Dizemos que $T(n)$ é $\Theta(f(n))$ 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 $\mathrm{O}(f(n))$
$T(n) \geq f(n)$	$T(n)$ 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
$20\, n \lceil \lg n \rceil$	4096	166666	7826087
$2n^2$	707	5477	42426
n^4	31	88	244
2^n	19	25	31

maior valor de n

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	$1{,}8\ 10^{19}$
128	7	11	896	16384	2097152	$3{,}4\ 10^{38}$
256	8	16	1048	65536	16777216	$1{,}1\ 10^{77}$
512	9	23	4608	262144	134217728	$1{,}3\ 10^{154}$
1024	10	32	10240	1048576	$1{,}1\ 10^{9}$	$1{,}7\ 10^{308}$

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

Palavras de Cautela

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

$100\,n$ é $\Theta(n)$ e $n \log_{10} n$ é $\Theta(n \lg n)$.
Logo, $\mathcal{A}$ é assintoticamente mais eficiente que $\mathcal{B}$.

Palavras de Cautela

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

$100\,n$ é $\Theta(n)$ e $n \log_{10} n$ é $\Theta(n \lg n)$.
Logo, $\mathcal{A}$ é assintoticamente mais eficiente que $\mathcal{B}$.

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

$$
\begin{aligned}
10^{100} &= \text{um } \textit{googol} \\
&\approx \text{número de átomos no universo observável} \\
&= \text{número ENORME}
\end{aligned}
$$

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 ← 2 até n faça
2      chave ← A[j]
3      i ← j − 1
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]        ▷ desloca
6          i ← i − 1
7      A[i + 1] ← chave        ▷ insere
```

Análise do caso médio

ORDENA-POR-INSERÇÃO (A, n)

1 **para** $j \leftarrow 2$ **até** n **faça**
2 $\quad chave \leftarrow A[j]$
3 $\quad i \leftarrow j - 1$
4 $\quad$ **enquanto** $i \geq 1$ **e** $A[i] > chave$ **faça**
5 $\quad\quad A[i+1] \leftarrow A[i]$ $\quad \triangleright$ desloca
6 $\quad\quad i \leftarrow i - 1$
7 $\quad A[i+1] \leftarrow chave$ $\quad \triangleright$ 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 ← 2 até n faça
2      chave ← A[j]
3      i ← j − 1
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]      ▷ desloca
6          i ← i − 1
7      A[i + 1] ← 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 ← 2 até n faça
2      chave ← A[j]
3      i ← j − 1
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]        ▷ desloca
6          i ← i − 1
7      A[i + 1] ← 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

formula da esperança

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

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 crescente se $A[1] \leq \cdots \leq A[n]$.

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

Ordenação

$A[1\,..\,n]$ é crescente se $A[1] \leq \cdots \leq A[n]$.

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

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

Ordenação

$A[1\,..\,n]$ é crescente se $A[1] \leq \cdots \leq A[n]$.

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

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

Conseguimos fazer melhor?

Ordenação

$A[1\,..\,n]$ é crescente se $A[1] \leq \cdots \leq A[n]$.

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

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

Conseguimos fazer melhor?

Sim! Usando divisão e conquista!