Melhores momentos

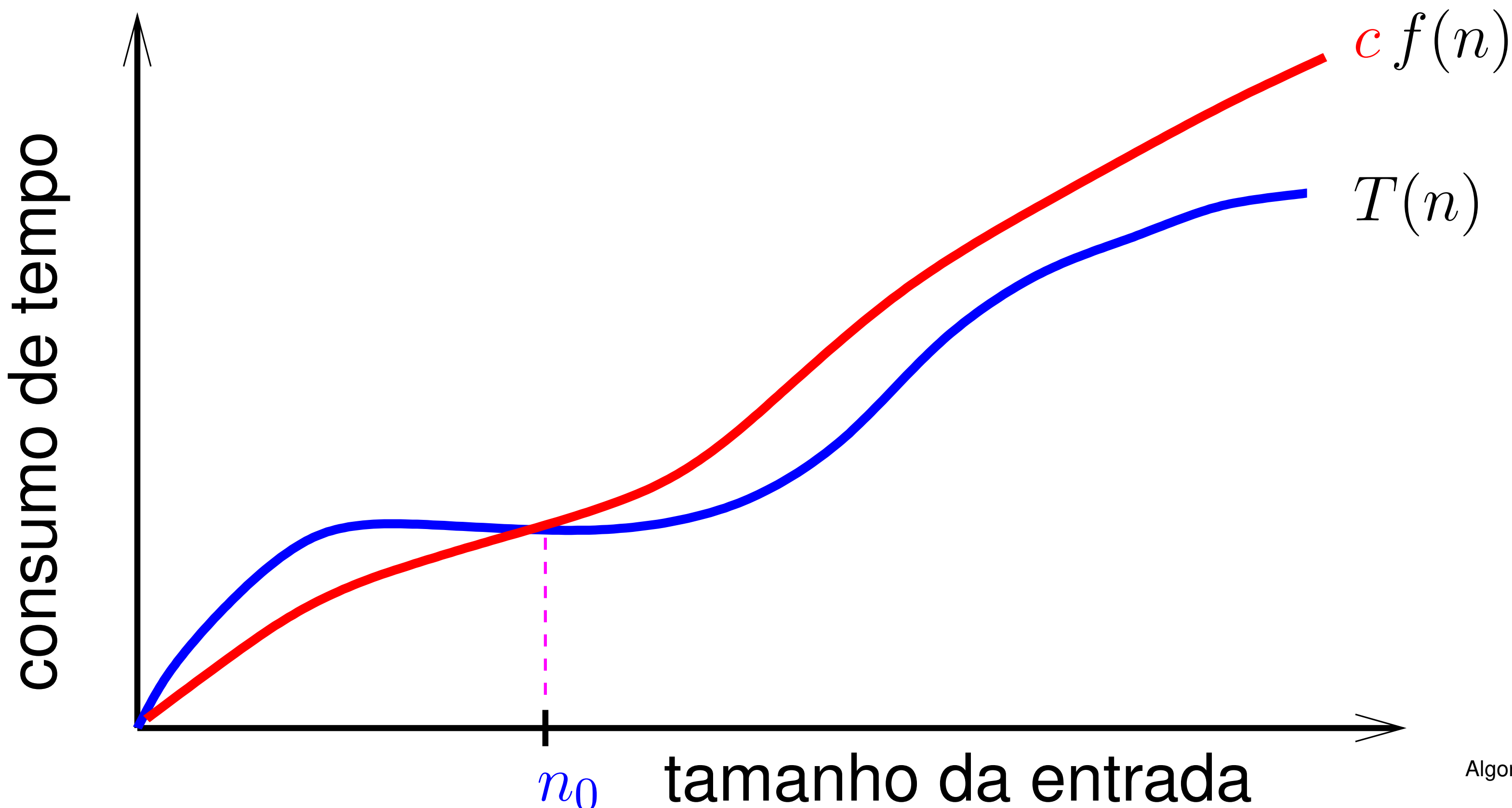
AULA PASSADA

Definição

Sejam $T(n)$ e $f(n)$ funções dos inteiros no reais. Dizemos que $T(n)$ **é** $\mathrm{O}(f(n))$ se existem constantes positivas c e n_0 tais que

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

para todo $n \geq n_0$.



Notação O

Intuitivamente...

$$\begin{array}{lcl} O(f(n)) & \approx & \text{funções q não crescem mais rápido que } f(n) \\ & \approx & \text{funções menores ou iguais a um múltiplo de } f(n) \end{array}$$

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

crescem todas com a mesma velocidade, são todas $O(n^2)$.

- $n^2 + 100n \lg n$ é $O(n^2)$
- $n \lg n + 33n$ é $O(n \lg n)$
- $0{,}00001n^3 - 200n^2$ **não é** $O(n^2)$
- 10^{100} é $O(1)$

Uso da notação O

$\mathrm{O}(f(n)) = \{T(n)$: existem c e n_0 tq $T(n) \leq cf(n), n \geq n_0\}$

"$T(n)$ é $\mathrm{O}(f(n))$" deve ser entendido como "$T(n) \in \mathrm{O}(f(n))$".

"$T(n) = \mathrm{O}(f(n))$" deve ser entendido como "$T(n) \in \mathrm{O}(f(n))$".

"$T(n) \leq \mathrm{O}(f(n))$" é feio.

"$T(n) \geq \mathrm{O}(f(n))$" não faz sentido!

"$T(n)$ é $g(n) + \mathrm{O}(f(n))$" significa que existem constantes positivas c e n_0 tais que

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

para todo $n \geq n_0$.

Análise da intercalação

Problema: Dados $A[p\,..\,q]$ e $A[q{+}1\,..\,r]$ crescentes, rearranjar $A[p\,..\,r]$ de modo que ele fique em ordem crescente.

Para que valores de q o problema faz sentido?

Entra:

$$
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
 & p & & & & q & & & & r \\
\cline{2-10}
A & 22 & 33 & 55 & 77 & 99 & 11 & 44 & 66 & 88 \\
\cline{2-10}
\end{array}
$$

Sai:

$$
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
 & p & & & & q & & & & r \\
\cline{2-10}
A & 11 & 22 & 33 & 44 & 55 & 66 & 77 & 88 & 99 \\
\cline{2-10}
\end{array}
$$

Intercalação

	p				q				r
A	22	33	55	77	99	11	44	66	88

B									

Intercalação

	k								
A									

	i								j
B	22	33	55	77	99	88	66	44	11

Intercalação

		k							
A	11								

	i							j	
B	22	33	55	77	99	88	66	44	11

Intercalação

		k							
A	11	22							

		i						j	
B	22	33	55	77	99	88	66	44	11

Intercalação

				k					
A	11	22	33						

			i					j	
B	22	33	55	77	99	88	66	44	11

Intercalação

					k				
A	11	22	33	44					

			i				j		
B	22	33	55	77	99	88	66	44	11

Intercalação

						k			
A	11	22	33	44	55				

				i			j		
B	22	33	55	77	99	88	66	44	11

Intercalação

							k		
A	11	22	33	44	55	66			

				i		j			
B	22	33	55	77	99	88	66	44	11

Intercalação

								k	
A	11	22	33	44	55	66	77		

					i	j			
B	22	33	55	77	99	88	66	44	11

Intercalação

									k
A	11	22	33	44	55	66	77	88	

					$i = j$				
B	22	33	55	77	99	88	66	44	11

Intercalação

A	11	22	33	44	55	66	77	88	99

					j	i			
B	22	33	55	77	99	88	66	44	11

Intercalação

```
INTERCALA (A, p, q, r)
    ▷ B[p . . r] é um vetor auxiliar
    para i ← p até q faça
        B[i] ← A[i]
    para j ← q + 1 até r faça
        B[r + q + 1 − j] ← A[j]
    i ← p
    j ← r
    para k ← p até r faça
        se B[i] ≤ B[j]
            então A[k] ← B[i]
                  i ← i + 1
            senão A[k] ← B[j]
                  j ← j − 1
```

Consumo de tempo

Quanto tempo consome em função de $n := r - p + 1$ (tamanho do problema)?

linha	todas as execuções da linha
1	$= \quad q - p + 2 \;=\; n - r + q + 1$
2	$= \quad q - p + 1 \;=\; n - r + q$
3	$= \quad r - (q + 1) + 2 \;=\; n - q + p$
4	$= \quad r - (q + 1) + 1 \;=\; n - q + p - 1$
5	$= \quad 1$
6	$= \quad 1$
7	$= \quad r - p + 2 \;=\; n + 1$
8	$= \quad r - p + 1 \;=\; n$
9–12	$= \quad 2\,(r - p + 1) \;=\; 2n$
total	$= \quad 8n - 2(r - p + 1) + 5 = \; 6n + 5$

Conclusão

O algoritmo INTERCALA consome $6n+5$ unidades de tempo.

Consumo de tempo em O

Quanto tempo consome em função de $n := r - p + 1$ (tamanho do problema)?

linha	consumo de todas as execuções da linha
1–4	$\mathrm{O}(n)$
5–6	$\mathrm{O}(1)$
7	$n\mathrm{O}(1) = \mathrm{O}(n)$
8	$n\mathrm{O}(1) = \mathrm{O}(n)$
9–12	$n\mathrm{O}(1) = \mathrm{O}(n)$
total	$\mathrm{O}(4n + 1)$ = $\mathrm{O}(n)$

Conclusão

O algoritmo INTERCALA consome $\mathrm{O}(n)$ unidades de tempo.

Também escreve-se

O algoritmo INTERCALA consome tempo $\mathrm{O}(n)$.

$n\mathrm{O}(1) = \mathrm{O}(n)$

"$n\mathrm{O}(1) = \mathrm{O}(n)$" significa "$n\mathrm{O}(1) \subset \mathrm{O}(n)$".

Ou seja, se $T(n)$ é $\mathrm{O}(1)$, então $nT(n)$ é $\mathrm{O}(n)$.

Prova: De fato, se $T(n)$ é $\mathrm{O}(1)$ então existem constantes c e n_0, tais que

$$T(n) \;\leq\; c \times 1$$

para todo $n \geq n_0$. Desta forma,

$$n\,T(n) \;\leq\; n\,c \times 1 = c\,n$$

para todo $n \geq n_0$. Logo $n\,T(n)$ é $\mathrm{O}(1)$.

AULA 3

Irmãos de O

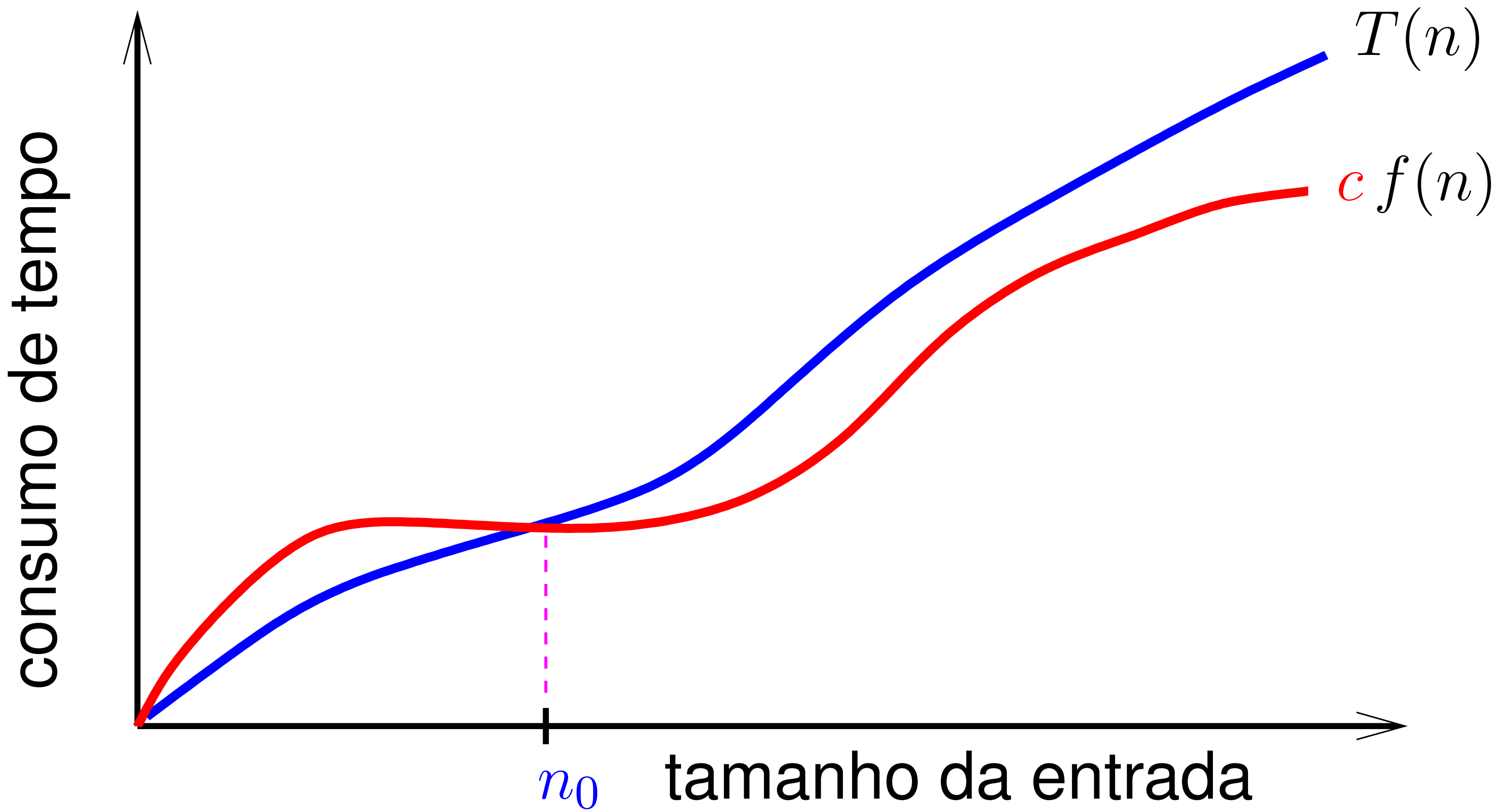
CLRS 3.1

Definição

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 $n \geq n_0$.

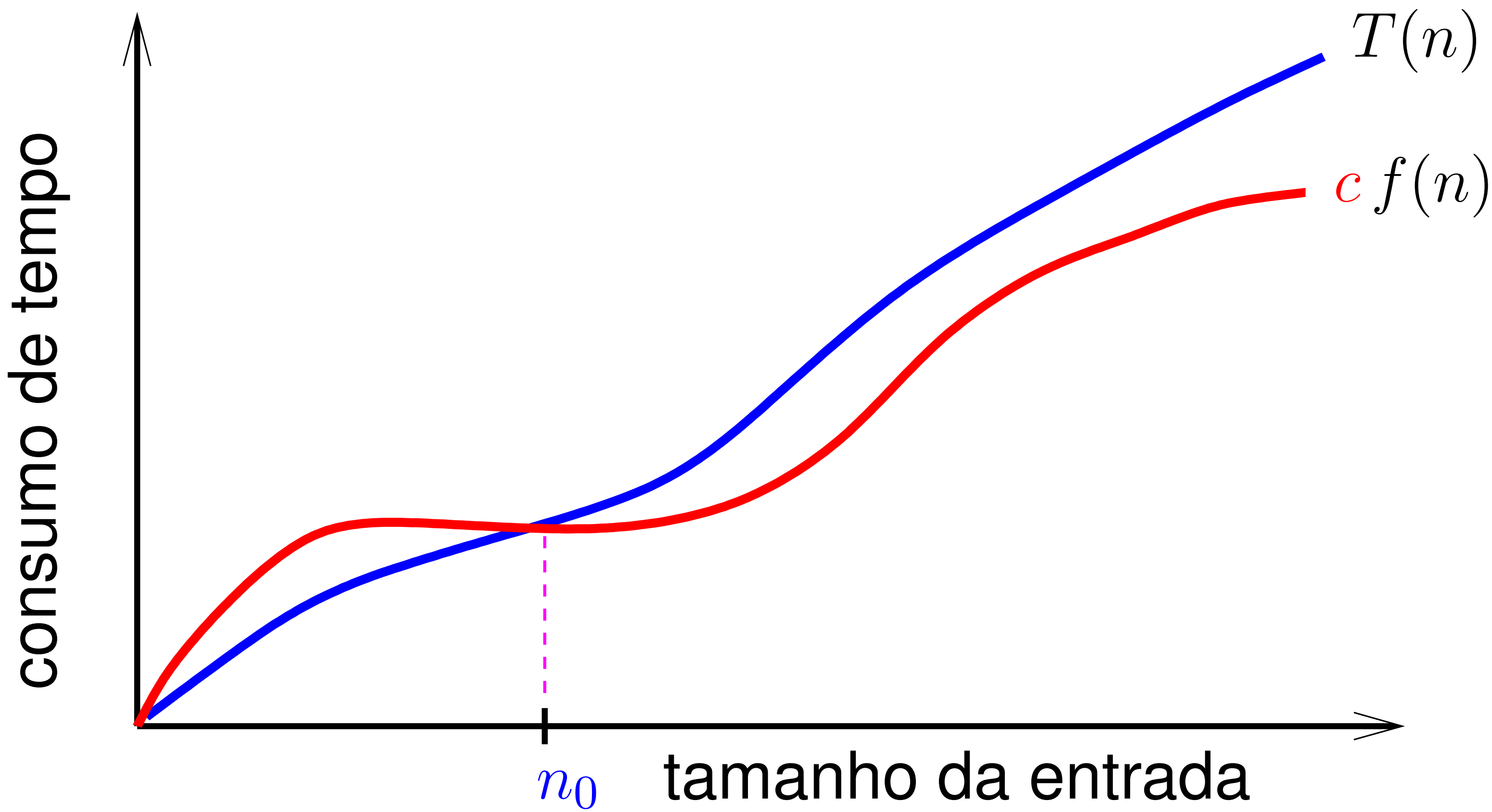


Mais informal

$T(n) = \Omega(f(n))$ 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 INTERCALA é $\mathrm{O}(n)$ e também $\Omega(n)$.

Exemplo 2

O consumo de tempo do INTERCALA é $\mathrm{O}(n)$ e também $\Omega(n)$.

linha		todas as execuções da linha
1	$=$	$q - p + 2 \;=\; n - r + q + 1$
2	$=$	$q - p + 1 \;=\; n - r + q$
3	$=$	$r - (q + 1) + 2 \;=\; n - q + p$
4	$=$	$r - (q + 1) + 1 \;=\; n - q + p - 1$
5	$=$	1
6	$=$	1
7	$=$	$r - p + 2 \;=\; n + 1$
8	$=$	$r - p + 1 \;=\; n$
9–12	$=$	$2\,(r - p + 1) \;=\; 2n$
total	$=$	$8n - 2(r - p + 1) + 5 = \; 6n + 5$

Exemplo 3

Se $T(n)$ é $\Omega(f(n))$, então $f(n)$ é $\mathrm{O}(T(n))$.

Exemplo 3

Se $T(n)$ é $\Omega(f(n))$, então $f(n)$ é $\mathrm{O}(T(n))$.

Prova: Se $T(n)$ é $\Omega(f(n))$, então existem constantes positivas c e n_0 tais que

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

para todo $n \geq n_0$. Logo,

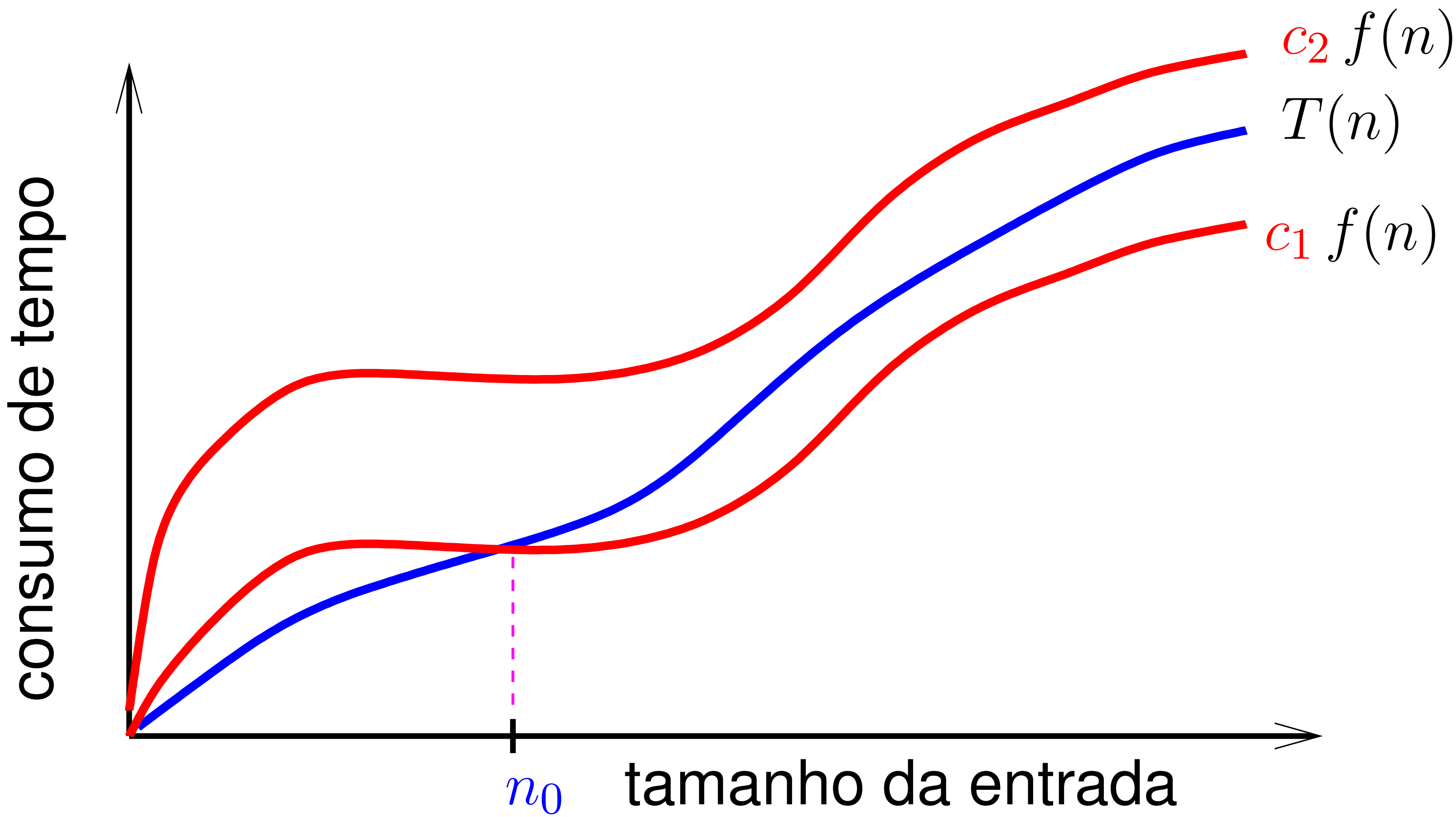
$$f(n) \;\leq\; 1/c\,T(n)$$

para todo $n \geq n_0$. Portanto, $f(n)$ é $\mathrm{O}(T(n))$.

Definição

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

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

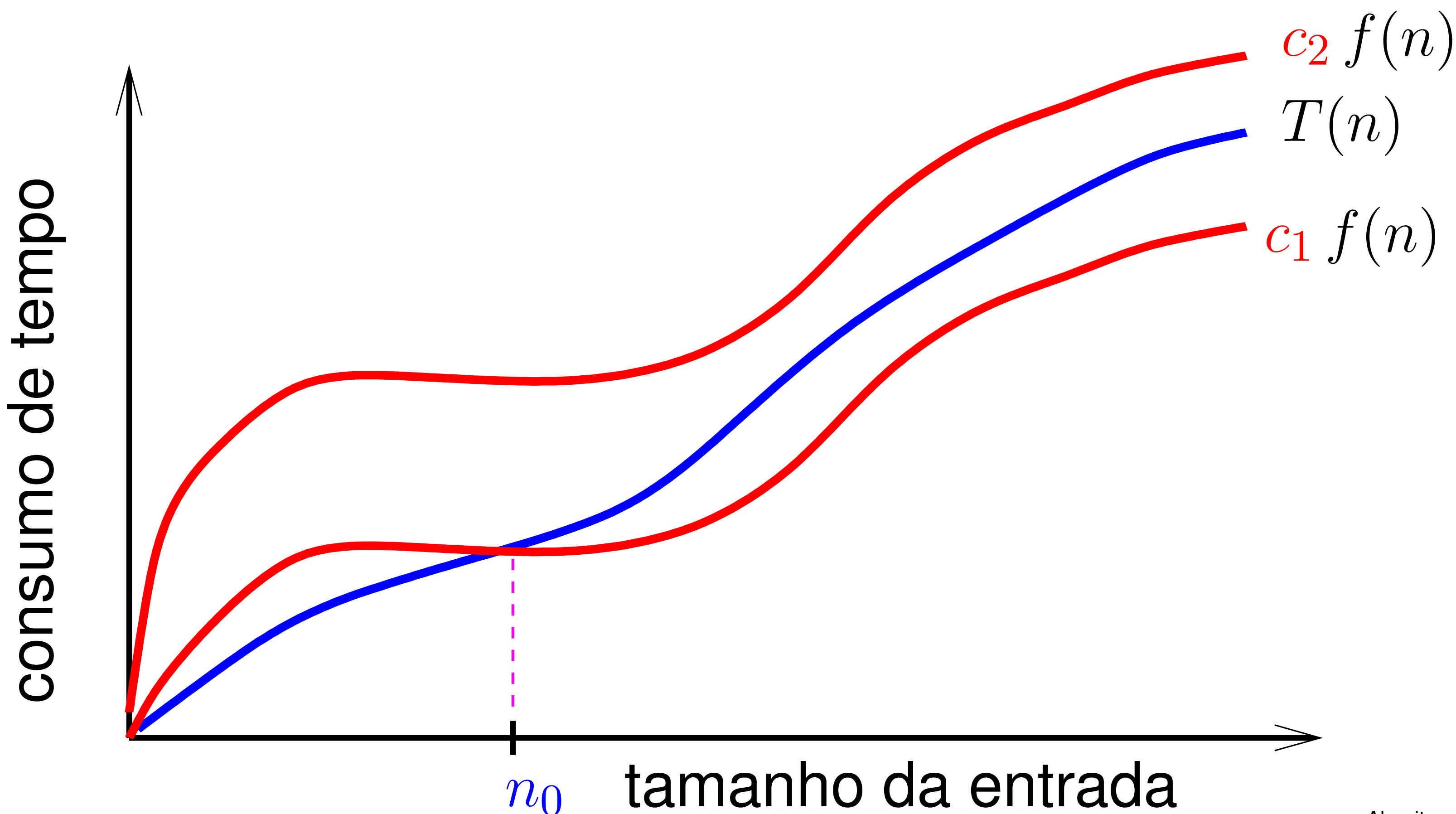


Definição

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 $\mathrm{O}(f(n))$
$T(n) \geq f(n)$	$T(n)$ 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

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 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.

Exercícios

Exercício 6.A
Já sabemos que ORDENA-POR-INSERÇÃO é $\mathrm{O}(n^2)$.
Mostre que o algoritmo é $\Omega(n)$.

Exercício 6.B
Mostre que ORDENA-POR-INSERÇÃO é $\Omega(n^2)$
no pior caso.

Exercício 6.C
Mostre que ORDENA-POR-INSERÇÃO é $\mathrm{O}(n)$
no melhor caso.

Mais exercícios

Exercício 6.D
Prove que $n^2 + 10n + 20 = \Omega(n^2)$. Prove que $n^2 - 10n - 20 = \Theta(n^2)$.

Exercício 6.E
Prove que $n = \Omega(\lg n)$.

Exercício 6.F
Prove que $\lg n = \Theta(\log_{10} n)$.

Exercício 6.G
É verdade que $2^n = \Omega(3^n)$?

Exercício 6.H
É verdade que $2n^3 + 5\sqrt{n} = \Theta(n^3)$?

Mais exercícios ainda

Exercício 6.I

Suponha que os algoritmos $\mathcal{A}$ e $\mathcal{B}$ só dependem de um parâmetro n. Suponha ainda que $\mathcal{A}$ consome $S(n)$ unidades de tempo enquanto $\mathcal{B}$ consome $T(n)$ unidades de tempo. Quero provar que algoritmo $\mathcal{A}$ é pelo menos tão eficiente quanto o algoritmo $\mathcal{B}$ (no sentido assintótico). Devo mostrar que existe $f(n)$ tal que

$S(n) = \mathrm{O}(f(n))$ e $T(n) = \mathrm{O}(f(n))$?

$S(n) = \mathrm{O}(f(n))$ e $T(n) = \Omega(f(n))$?

$S(n) = \Omega(f(n))$ e $T(n) = \mathrm{O}(f(n))$?

$S(n) = \Omega(f(n))$ e $T(n)\Omega(f(n))$?

Que devo fazer para mostrar que $\mathcal{A}$ é mais eficiente que $\mathcal{B}$?

Exercício 6.J

Mostre que o consumo de tempo do algoritmo INTERCALA é $\Theta(n)$, sendo n o número de elementos do vetor que o algoritmo recebe.

Recursão

CLRS 2.3
AU 2.6, 2.7, 2.9

"To understand recursion, we must first understand recursion.”

Recursão

Recursão: resolve um problema a partir das soluções de seus subproblemas

Recursão

Recursão: resolve um problema a partir das soluções de seus subproblemas

Problema: Rearranjar $A[p \,..\, r]$ de modo que ele fique em ordem crescente.

Entra:

$$
\begin{array}{c|c|c|c|c|c|c|c|c|c|}
 & p & & & & & & & & r \\
\cline{2-10}
A & 55 & 33 & 66 & 44 & 99 & 11 & 77 & 22 & 88 \\
\cline{2-10}
\end{array}
$$

Recursão

Recursão: resolve um problema a partir das soluções de seus subproblemas

Problema: Rearranjar $A[p \mathinner{.\,.} r]$ de modo que ele fique em ordem crescente.

Entra:

	p								r
A	55	33	66	44	99	11	77	22	88

Sai:

	p								r
A	11	22	33	44	55	66	77	88	99

Divisão-e-conquista

Algoritmos por divisão-e-conquista têm três passos em cada nível da recursão:

Dividir: o problema é dividido em subproblemas de tamanho menor;

Conquistar: os subproblemas são resolvidos recursivamente e subproblemas "pequenos" são resolvidos diretamente;

Combinar: as solução dos subproblemas são combinadas para obter uma solução do problema original.

Exemplo: ordenação por intercalação (Merge-sort).

Merge-Sort

	p				q				r
A	55	33	66	44	99	11	77	22	88

Merge-Sort

	p				q				r
A	55	33	66	44	99	11	77	22	88

	p		q		r				
A	55	33	66	44	99				

Merge-Sort

	p				q				r
A	55	33	66	44	99	11	77	22	88

	p		q		r				
A	55	33	66	44	99				

	p	q	r						
A	55	33	66						

Merge-Sort

	p				q				r
A	55	33	66	44	99	11	77	22	88

	p		q		r				
A	55	33	66	44	99				

	p	q	r						
A	55	33	66						

	p	r							
A	55	33							

Merge-Sort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

	p	q	r						
A	33	55	66						

	p	r							
A	33	55							

Merge-Sort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

	p	q	r						
A	33	55	66						

			$p = r$						
A			66						

Merge-Sort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

	p	q	r						
A	33	55	66						

Merge-Sort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

Merge-Sort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

				p	r				
A				44	99				

Merge-Sort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

				p	r				
A				44	99				

Merge-Sort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

	p		q		r				
A	33	44	55	66	99				

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p			r
A						11	77	22	88

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p			r
A						11	77	22	88

						p	r		
A						11	77		

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p			r
A						11	77	22	88

						p	r		
A						11	77		

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p			r
A						11	77	22	88

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p			r
A						11	77	22	88

								p	r
A								22	88

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p			r
A						11	77	22	88

								p	r
A								22	88

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p			r
A						11	77	22	88

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	22	77	88

						p			r
A						11	22	77	88

Merge-Sort

	p				q				r
A	33	44	55	66	99	11	22	77	88

Merge-Sort

	p				q				r
A	11	22	33	44	55	66	77	88	99

Merge-Sort

	p				q				r
A	11	22	33	44	55	66	77	88	99

Merge-Sort

Rearranja $A[p \,..\, r]$, com $p \leq r$, em ordem crescente.

MERGE-SORT (A, p, r)
    se $p < r$
        então $q \leftarrow \lfloor (p + r)/2 \rfloor$
                MERGE-SORT (A, p, q)
                MERGE-SORT $(A, q + 1, r)$
                INTERCALA (A, p, q, r)

	p				q				r
A	55	33	66	44	99	11	77	22	88

Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
    se p < r
        então q ← ⌊(p + r)/2⌋
              MERGE-SORT (A, p, q)
              MERGE-SORT (A, q + 1, r)
              INTERCALA (A, p, q, r)
```

	p				q				r
A	33	44	55	66	99	11	77	22	88

Merge-Sort

Rearranja $A[p \,..\, r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
    se p < r
        então q ← ⌊(p + r)/2⌋
              MERGE-SORT (A, p, q)
              MERGE-SORT (A, q + 1, r)
              INTERCALA (A, p, q, r)
```

	p				q				r
A	33	44	55	66	99	11	22	77	88

Merge-Sort

Rearranja $A[p \mathinner{..} r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
    se p < r
        então q ← ⌊(p + r)/2⌋
            MERGE-SORT (A, p, q)
            MERGE-SORT (A, q + 1, r)
            INTERCALA (A, p, q, r)
```

	p				q				r
A	11	22	33	44	55	66	77	88	99

Merge-Sort

Rearranja $A[p \mathinner{..} r]$, com $p \leq r$, em ordem crescente.

MERGE-SORT (A, p, r)
se $p < r$
  então $q \leftarrow \lfloor (p + r)/2 \rfloor$
    MERGE-SORT (A, p, q)
    MERGE-SORT $(A, q + 1, r)$
    INTERCALA (A, p, q, r)

O algoritmo está correto?

Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

MERGE-SORT (A, p, r)
1 **se** $p < r$
2 **então** $q \leftarrow \lfloor (p+r)/2 \rfloor$
3 MERGE-SORT (A, p, q)
4 MERGE-SORT $(A, q+1, r)$
5 INTERCALA (A, p, q, r)

O algoritmo está correto?
A correção do algoritmo, que se apóia na correção do INTERCALA, pode ser demonstrada por indução em $n := r - p + 1$.

Merge-Sort

Rearranja $A[p \mathinner{..} r]$, com $p \leq r$, em ordem crescente.

MERGE-SORT (A, p, r)
1 **se** $p < r$
2 **então** $q \leftarrow \lfloor (p + r)/2 \rfloor$
3 MERGE-SORT (A, p, q)
4 MERGE-SORT $(A, q + 1, r)$
5 INTERCALA (A, p, q, r)

Consumo de tempo?

Merge-Sort

Rearranja $A[p \mathinner{..} r]$, com $p \leq r$, em ordem crescente.

```
MERGE-SORT (A, p, r)
    se p < r
        então q ← ⌊(p + r)/2⌋
              MERGE-SORT (A, p, q)
              MERGE-SORT (A, q + 1, r)
              INTERCALA (A, p, q, r)
```

Consumo de tempo?

$T(n) :=$ consumo de tempo **máximo** quando $n = r - p + 1$

Merge-Sort

```
MERGE-SORT (A, p, r)
1    se p < r
2        então q ← ⌊(p + r)/2⌋
3            MERGE-SORT (A, p, q)
4            MERGE-SORT (A, q + 1, r)
5            INTERCALA (A, p, q, r)
```

linha	consumo na linha
1	?
2	?
3	?
4	?
5	?

$T(n) \quad = \quad ?$

Merge-Sort

```
MERGE-SORT (A, p, r)
1   se p < r
2       então q ← ⌊(p + r)/2⌋
3           MERGE-SORT (A, p, q)
4           MERGE-SORT (A, q + 1, r)
5           INTERCALA (A, p, q, r)
```

linha	consumo na linha
1	$\Theta(1)$
2	$\Theta(1)$
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	$\Theta(n)$

$$T(n) \quad = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n+2)$$

Merge-Sort

$T(n) :=$ consumo de tempo máximo quando $n = r - p + 1$

$$T(1) = \Theta(1)$$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \quad \text{para } n = 2, 3, 4, \ldots$$

Merge-Sort

$T(n)$:= consumo de tempo máximo quando $n = r - p + 1$

$$T(1) = \Theta(1)$$
$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \quad \text{para } n = 2, 3, 4, \ldots$$

Solução: $T(n)$ é $\Theta(???)$.

Demonstração: ...

Merge-Sort

$T(n) :=$ consumo de tempo máximo quando $n = r - p + 1$

$$T(1) = \Theta(1)$$
$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \quad \text{para } n = 2, 3, 4, \ldots$$

Solução: $T(n)$ é $\Theta(???)$.

Demonstração: $\ldots$

Veremos, mas antes estudaremos Recorrências.

Exercícios

Exercício 7.A
Problema: verificar se v é elemento de $A[p \mathinner{.\,.} r]$.
(Para quais valores de p e r faz sentido?)
Escreva um algoritmo recursivo que resolve o problema. O algoritmo deve devolver i tal que $A[i] = v$.

Exercício 7.B
Problema: verificar se v é elemento de vetor crescente $A[p \mathinner{.\,.} r]$. Escreva algoritmo recursivo de busca "linear" e outro de busca "binária".

Exercício 7.C
Escreva versão recursiva da ordenação por inserção. O algoritmo deve rearranjar em ordem crescente qualquer vetor dado $A[p \mathinner{.\,.} r]$.

Mais exercícios

Exercício 7.D (Versão sofisticada de busca)
Problema: verificar se v é elemento de vetor crescente $A[p\,..\,r]$. Escreva um algoritmo que devolva j tal que

$$A[j] \leq v < A[j+1]\ .$$

Quais os possíveis valores de j? Escreva duas versões: uma "linear" e uma "binária". Prove que os seus algoritmos estão corretos.

Exercício 7.E
Escreva uma versão recursiva do algoritmo de ordenação por seleção.

Exercício 7.F
Escreva uma versão iterativa do MERGE-SORT.

Recorrências

CLRS 4.1–4.2

AU 3.9, 3.11

Recorrências

Recorrência =

= "fórmula" que define uma função
em termos d'ela mesma

= algoritmo recursivo que calcula uma função

Exemplo 1

$$
\begin{aligned}
T(1) &= 1 \\
T(n) &= T(n-1) + 3n + 2 \quad \text{para } n = 2, 3, 4, \ldots
\end{aligned}
$$

Define função T sobre inteiros positivos:

n	1	2	3	4	5	6
$T(n)$	1	9	20	34	51	71

$T(n)$
se $n = 1$
  então devolva 1
  senão devolva $T(n-1) + 3n + 2$

Resolver uma recorrência

Resolver uma recorrência =

= obter uma “fórmula fechada” para $T(n)$

Método da substituição:

“chute” fórmula e verifique por indução

Exemplo 1 (continuação)

Eu acho que $T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4$.

Exemplo 1 (continuação)

Eu acho que $T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4$.

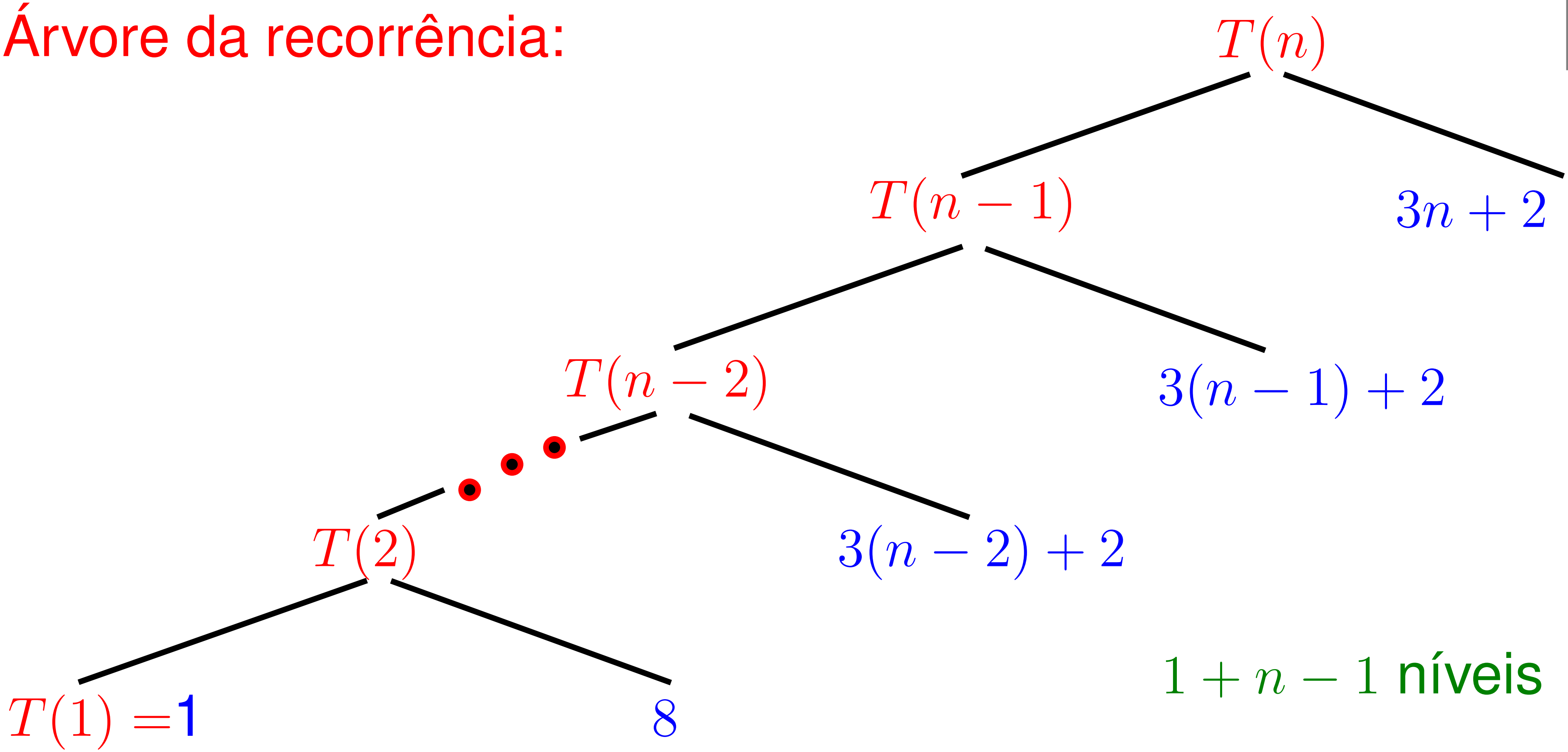
Verificação:
Se $n = 1$ então $T(n) = 1 = \frac{3}{2} + \frac{7}{2} - 4$.
Tome $n \geq 2$ e suponha que a fórmula está certa para $n - 1$:

$$
\begin{aligned}
T(n) &= T(n-1) + 3n + 2 \\
&\overset{\text{hi}}{=} \frac{3}{2}(n-1)^2 + \frac{7}{2}(n-1) - 4 + 3n + 2 \\
&= \frac{3}{2}n^2 - 3n + \frac{3}{2} + \frac{7}{2}n - \frac{7}{2} - 4 + 3n + 2 \\
&= \frac{3}{2}n^2 + \frac{7}{2}n - 4\,.
\end{aligned}
$$

Bingo!

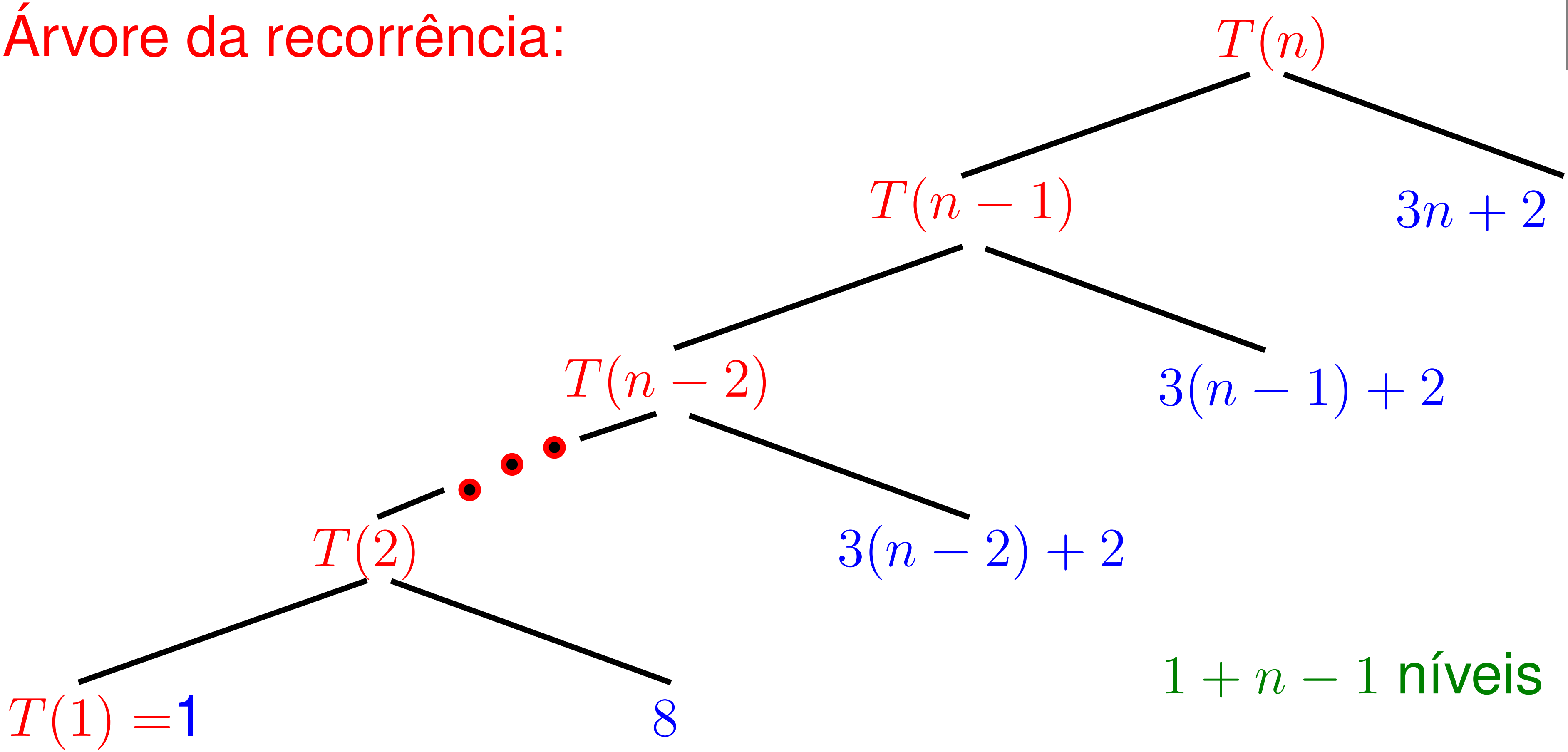
Como adivinhei fórmula fechada?

Árvore da recorrência:



Como adivinhei fórmula fechada?

Árvore da recorrência:



$$T(n) = (3n+2) + (3n-1) + \cdots + 8 + 1$$

$$= \frac{3}{2}n^2 + \frac{7}{2}n - 4$$

Exemplo 2

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(n/2) + 7n + 2 \quad \text{para } n = 2, 3, 4, 5, \ldots \end{aligned}$$

Exemplo 2

$$
\begin{aligned}
T(1) &= 1 \\
T(n) &= 2T(n/2) + 7n + 2 \quad \text{para } n = 2, 3, 4, 5, \ldots
\end{aligned}
$$

Não é uma recorrência! Não faz sentido!
$T(3)$ depende de $T(3/2)$...

Exemplo 3

$$\begin{aligned} G(1) &= 1 \\ G(n) &= 2G(n/2) + 7n + 2 \quad \text{para } n = 2, 4, 8, 16\,.\,.\,, 2^i, \,.\,. \end{aligned}$$

n	1	2	4	8	16
$G(n)$	1	18	66	190	494

Exemplo 3

$$
\begin{aligned}
G(1) &= 1 \\
G(n) &= 2G(n/2) + 7n + 2 \quad \text{para } n = 2, 4, 8, 16 \ldots, 2^i, \ldots
\end{aligned}
$$

n	1	2	4	8	16
$G(n)$	1	18	66	190	494

Fórmula fechada: $G(n) =$???