Análise de Algoritmos

Ronaldo Cristiano Prati

Bloco A, sala 513-2

ronaldo.prati@ufabc.edu.br

Resolução de exercícios

- Dado que $f(n) = \log \sqrt{n}$ e $g(n) = \log(100n)$, mostre que $f(n) = O(g(n))$

$$\begin{aligned} \log \sqrt{n} &\leq c \log(100n) \\ \log n^{1/2} &\leq c \log(100n) \\ \frac{1}{2} \times \frac{\log n}{\log(100n)} &\leq c \\ \frac{1}{2} \times \frac{\log n}{\log(100) + \log(n)} &\leq c \end{aligned}$$

- Observe que para $n \geq 1$, o termo $\frac{\log n}{\log(100)+\log(n)}$ está constrito entre 0 e 1, portanto, podemos escolher $c > 1/2$ para um $n_0 \geq 1$

Resolução de exercícios

- Dado que $f(n) = n!$ e $g(n) = n \log n$, mostre que $f(n)$ não é $O(g(n))$
- Vamos provar por absurdo.

$$
\begin{aligned}
n! &\leq cn \log n \\
n(n-1)(n-2)! &\leq cn \log n \\
n(n-1)(n-2)! &\leq cn(n-1) \\
(n-2)! &\leq c
\end{aligned}
$$

- O quer claramente não é possível para $n_0 > 2$, uma vez que $n!$ (e por consequência $(n-2)!$) é sempre crescente, e não pode ser limita por uma constante.

Resolução de exercícios

- Ordene as funções a seguir em ordem crescente de taxa de crescimento:

$$T_1 = 2^n$$
$$T_2 = n^{3/2}$$
$$T_3 = n \log n$$
$$T_4 = n^{\log n}$$

Resolução de exercícios

- Afirmação 1: T_3 é (assintoticamente) a menor delas, pois nenhuma delas é $O(T_3)$
- T_2 não é $O(T_3)$. Vamos assumir que é, então

$$n^{3/2} \leq cn \log n$$

$$\frac{n^{3/2}}{n \log n} \leq c$$

$$\frac{\sqrt{n}}{\log n} \leq c$$

- $\sqrt{n}$ é maior que $\log n$ para n grande ($\lim_{n \to \infty} \frac{n^p}{\log n} = \infty, \forall p > 0$), portanto não é possível ser limitada por uma constante

Resolução de exercícios

- Afirmação 1: T_3 é (assintoticamente) a menor delas, pois nenhuma delas é $O(T_3)$
- T_4 não é $O(T_3)$. Vamos assumir que é, então

$$n^{\log n} \leq cn \log n$$

$$\frac{n^{\log n}}{n \log n} \leq c$$

$$\frac{n^{\log n - 1}}{\log n} \leq c$$

- $n^{\log n - 1}$ é maior que $\log n$ para n grande, portanto não é possível ser limitada por uma constante

Resolução de exercícios

- Afirmação 1: T_3 é (assintoticamente) a menor delas, pois nenhuma delas é $O(T_3)$
- T_1 não é $O(T_3)$. Vamos assumir que é, então

$$2^n \leq cn \log n$$

$$\frac{2^n}{n \log n} \leq c$$

- 2^n é maior que $n \log n$ para n grande, portanto não é possível ser limitada por uma constante

Resolução de exercícios

- Afirmação 2: A segunda (assintoticamente) menor é T_2, pois T_4 e T_1 não são $O(T_2)$
- T_4 não é $O(T_2)$. Vamos assumir que é, então

$$
\begin{aligned}
n^{\log n} &\leq cn^{3/2} \\
\frac{n^{\log n}}{n^{3/2}} &\leq c \\
n^{\log n - 3/2} &\leq c
\end{aligned}
$$

- que não é possível ser limitada por uma constante

Resolução de exercícios

- Afirmação 2: A segunda (assintoticamente) menor é T_2, pois T_4 e T_1 não são $O(T_2)$
- T_1 não é $O(T_2)$. Vamos assumir que é, então

$$\begin{aligned} 2^n &\leq cn^{3/2} \\ \frac{2^n}{n^{3/2}} &\leq c \\ n - \frac{3}{2}\log n &\leq \log c \end{aligned}$$

- que não é possível ser limitada por uma constante

Resolução de exercícios

- Afirmação3: T_4 é (assintoticamente) menor que T_1, pois T_1 não é $O(T_4)$. Vamos assumir que é, então

$$\begin{aligned} 2^n &\leq cn^{\log n} \\ \frac{2^n}{n^{\log n}} &\leq c \\ n - \log n \log n &\leq \log c \end{aligned}$$

- Observe que $\log n \log n$ é menor que $\sqrt{n}\sqrt{n} = n$, então n cresce mais rápido que $\log n \log n$, e não é possível limitar por uma constante

Resolução de exercícios

- Juntanto as afirmações 1, 2 e 3 temos que (assintoticamente):

$$T_3 < T_2 < T_4 < T_1$$

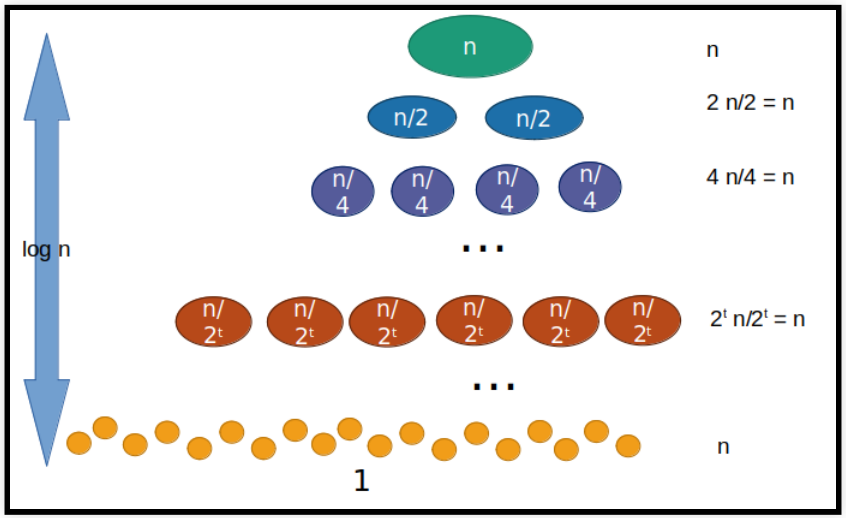
Árvore de recursão

- Árvore de recursão é um método muito útil para estimar a solução de uma recorrência
- Uma vez obtida a estimativa, podemos usar o método da substituição para prová-la
- A ideia consiste em representar o desenvolvimento da recorrência por meio de um diagrama (geralmente uma árvore), em que cada nó representa um subproblema, e somar os custos por nível

Árvore de recursão

- Considere por exemplo a relação de recorrência

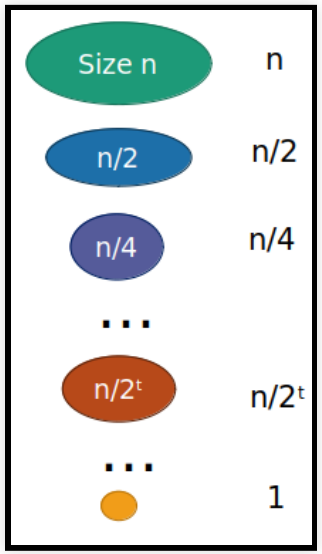
$$T(n) = 2T(n/2) + \Theta(n)$$



Árvore de recursão

- Considere por exemplo a relação de recorrência

$$T(n) = T(n/2) + \Theta(n)$$

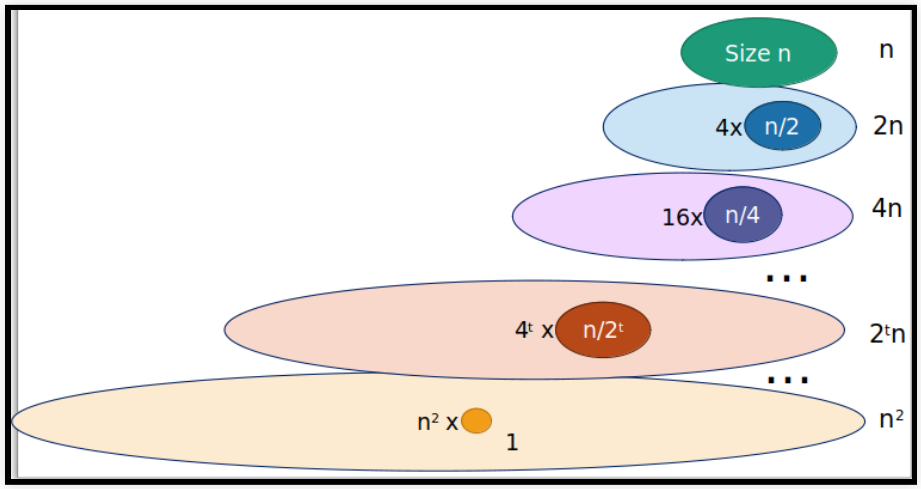


$$T(n) = \sum_{t=0}^{\log n} \frac{n}{2^t} = 2n - 1 = \Theta(n)$$

Árvore de recursão

- Considere por exemplo a relação de recorrência

$$T(n) = 4T(n/2) + \Theta(n)$$



$$T(n) = \sum_{t=0}^{\log n} 4^t \frac{n}{2^t} = n \sum_{t=0}^{\log n} 2^t = n(2n-1) = \Theta(n^2)$$

Método Mestre (simplificado)

- O método mestre provê uma fórmula que pode ser aplicada a equações de recorrência do tipo:

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d)$$

- em que:
  - a é o número de suproblemas
  - b fator em que o tamanho da entrada diminui
  - d é necessário fazer n^d trabalho adicional para criar os subproblemas e combinar as soluções

Método Mestre (simplificado)

- Seja:

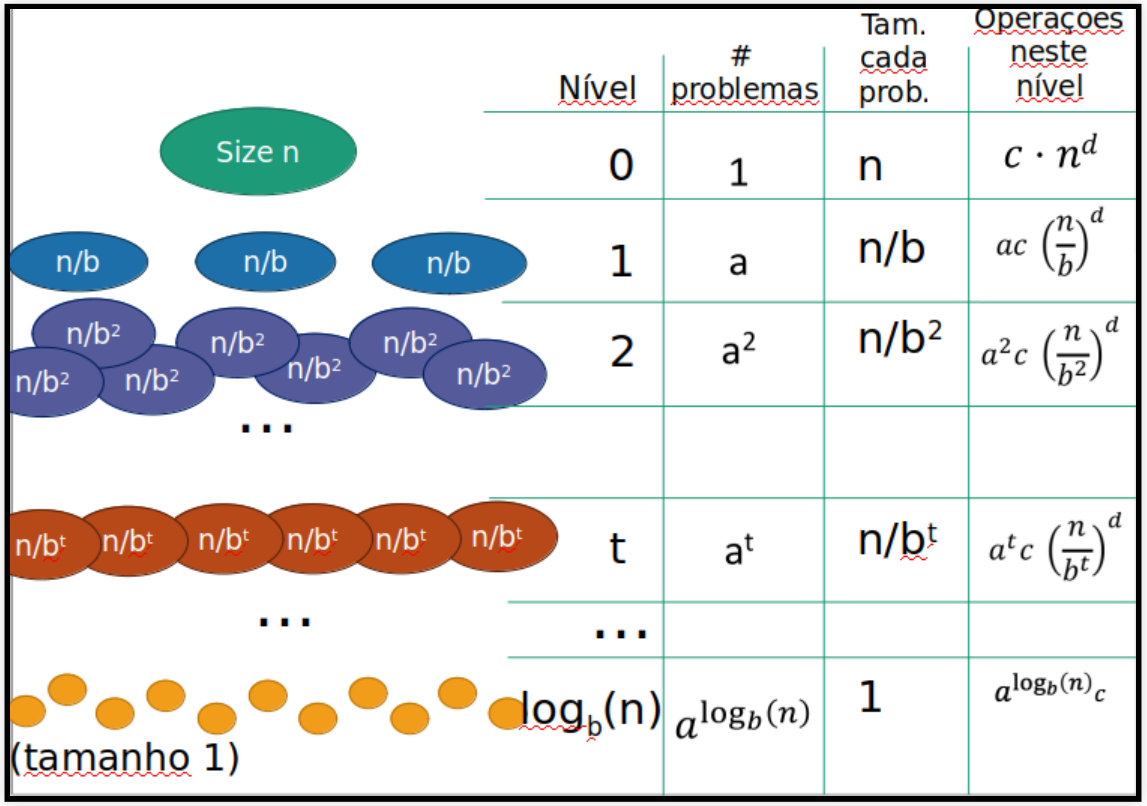
$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d)$$

- Então

$$T(n) = \begin{cases} \Theta\left(n^d \log(n)\right) & \text{se } a = b^d \\ \Theta\left(n^d\right) & \text{se } a < b^d \\ \Theta\left(n^{\log_b(a)}\right) & \text{se } a > b^d \end{cases}$$

Método Mestre (simplificado)

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d)$$



Nível	# problemas	Tam. cada prob.	Operações neste nível
0	1	n	$c \cdot n^d$
1	a	n/b	$ac\left(\frac{n}{b}\right)^d$
2	a^2	n/b^2	$a^2c\left(\frac{n}{b^2}\right)^d$
t	a^t	n/b^t	$a^tc\left(\frac{n}{b^t}\right)^d$
...			
$\log_b(n)$	$a^{\log_b(n)}$	1	$a^{\log_b(n)}c$

Método Mestre (simplificado)

- O número de computção no nível t é (no máixmo):

$$a_t \cdot c \cdot \left(\frac{n}{b^t}\right)^d = c \cdot n^d \cdot \left(\frac{a}{b^d}\right)^t$$

- Em que:
  - a_t: número de subproblemas no nível t
  - $\left(\frac{n}{b^t}\right)$: tamanho de cada suproblema
  - $c\left(\frac{n}{b^t}\right)^d$: a quatidade de trabalho para cada subproblema

Método Mestre (simplificado)

- Somando sobre todos os níveis temos que o total de trabalho é (no máximo):

$$T(n) = c \cdot n^d \cdot \sum_{t=0}^{\log_b n} \left(\frac{a}{b^d} \right)^t$$

Caso 1

- Nesse caso, $a = b^d$

$$
\begin{aligned}
T(n) &= c \cdot n^d \cdot \sum_{t=0}^{\log_b(n)} \left(\frac{a}{b^d}\right)^t \\
&= c \cdot n^d \cdot \sum_{t=0}^{\log_b(n)} 1 \\
&= c \cdot n^d \cdot (\log_b(n) + 1) \\
&= c \cdot n^d \cdot \left(\frac{\log(n)}{\log(b)} + 1\right) \\
&= \Theta\left(n^d \log(n)\right)
\end{aligned}
$$

Caso 2

- Nesse caso, $a < b^d$
- O termo $\frac{a}{b^d}$ é menor que 1, e $\sum_{t=0}^{\log_b(n)} \left(\frac{a}{b^d}\right)^t$ converge para alguma constante menor que 1 :

$$\sum_{i=0}^{N} x^t = \frac{x^{N+1} - 1}{x - 1} \leq \frac{1 - x^{N+1}}{1 - x} \leq \frac{1}{1 - x} = \Theta(1)$$

Caso 2

- Nesse caso, $a < b^d$
- O termo $\frac{a}{b^d}$ é menor que 1, e $\sum_{t=0}^{\log_b(n)} \left(\frac{a}{b^d}\right)^t$ converge para alguma constante menor que 1:

$$\begin{aligned} T(n) &= c \cdot n^d \cdot \sum_{t=0}^{\log_b(n)} \left(\frac{a}{b^d}\right)^t \\ &= c \cdot n^d \cdot (\text{alguma constate}) \\ &= \Theta(n^d) \end{aligned}$$

Caso 3

- Nesse caso, $a > b^d$
- O termo $\frac{a}{b^d}$ é maior que 1:

$$\sum_{i=0}^{N} x^t = \frac{x^{N+1} - 1}{x - 1} \leq \frac{1 - x^{N+1}}{1 - x} \leq x^N \left(\frac{x}{x - 1} \right) = \Theta(x^N)$$

Caso 3

- Nesse caso, $a > b^d$
- O termo $\frac{a}{b^d}$ é maior que 1:

$$
\begin{aligned}
T(n) =& c \cdot n^d \cdot \sum_{t=0}^{\log_b(n)} \left(\frac{a}{b^d}\right)^t \\
&= \Theta\left(n^d \left(\frac{a}{b^d}\right)^{\log_b(n)}\right) \\
&= \Theta\left(n^{\log_b(a)}\right)
\end{aligned}
$$

Método mestre

- Em uma recorrência, temos dois fatores conflitantes:
  - A divisão faz com que o número de problemas exploda!
    - A maior parte do trabalho é feita na parte inferior da árvore
  - Os problemas mais abaixo na árvore são menores
    - A maior parte do trabalho está no topo da árvore

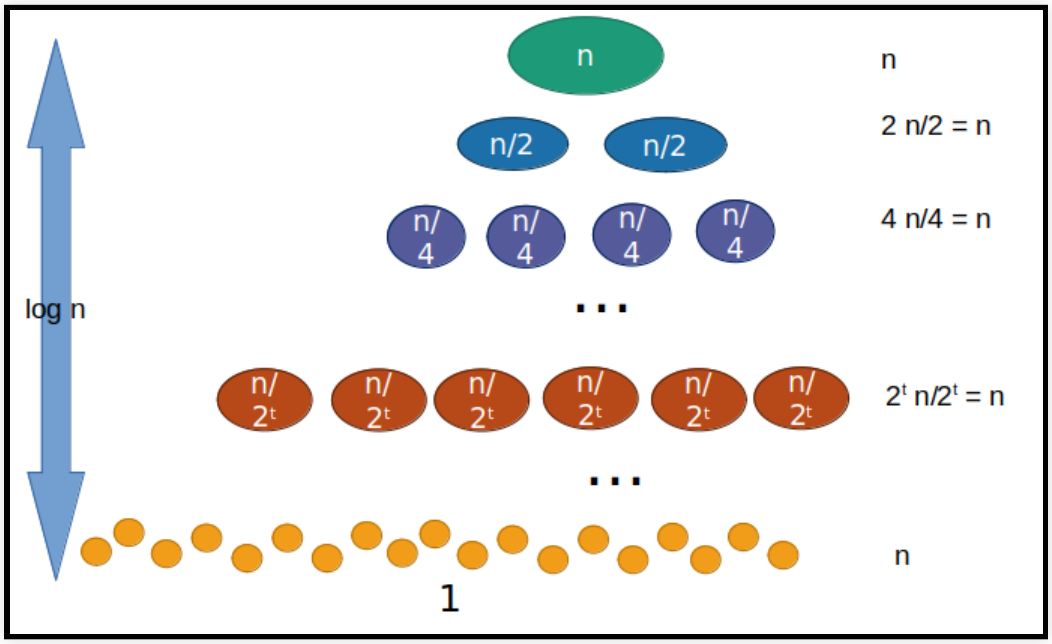
Método mestre

- Vamos considerar 3 exemplos para entender o método Mestre
  - $T_1(n) = 2T(n/2) + n$
  - $T_2(n) = T(n/2) + n$
  - $T_4(n) = 4T(n/2) + n$

Método mestre - Caso 1

- A quatidade de trabalho é similar em cada nivel, e a complexidade é proporcional ao número de níveis vezes o trabalho por nível (caso 1)

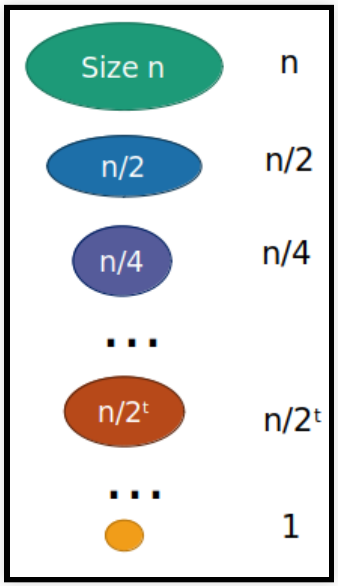
$$\begin{aligned} T_2(n) &= 2T(n/2) + n, a = b^d \\ &= \Theta(n^d \log n) = \Theta(n \log n) \end{aligned}$$


n
n
n/2
n/2
2 n/2 = n
n/4
n/4
n/4
n/4
4 n/4 = n
log n
...
n/2^t
n/2^t
n/2^t
n/2^t
n/2^t
n/2^t
2^t n/2^t = n
...
n
1

Método mestre - Caso 2

- A maioria do trabalho é feita no topo (o maior problema), que é maior que o trabalho em qualquer nível e domina a recorrência (caso 2)

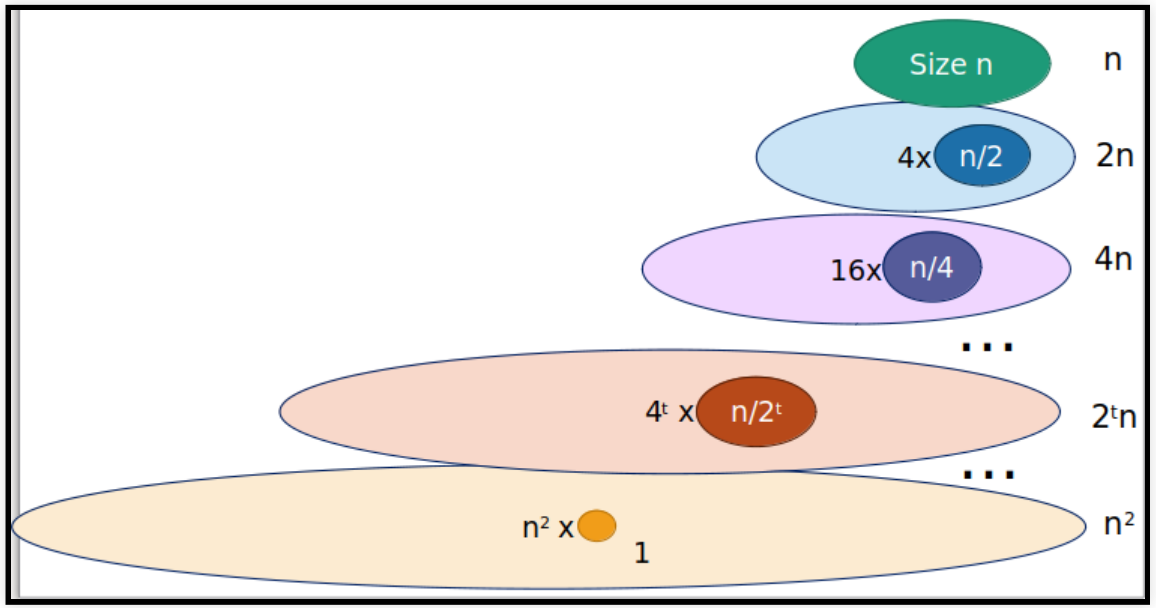
$$T_2(n) = T(n/2) + n, a < b^d$$
$$= \Theta(n^d) = \Theta(n)$$


Size n
n
n/2
n/2
n/4
n/4
...
n/2^t
n/2^t
...
1

Método mestre - Caso 3

- Há um grande número de folhas e o tempo é proporcional a processar as folhas (caso 3)

$$\begin{aligned} T_2(n) &= 4T(n/2) + n, a > b^d \\ &= \Theta(n^{\log_2 4}) = \Theta(n^2) \end{aligned}$$


Size n
n
4x
n/2
2n
16x
n/4
4n
...
4^t x
n/2^t
2^t n
...
n^2 x
1
n^2

Método Mestre (generalizado)

- O método mestre provê uma fórmula que pode ser aplicada a equações de recorrência do tipo:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- em que:
  - a é o número de suproblemas
  - b fator em que o tamanho da entrada diminui
  - $f(n)$ uma função

Método Mestre (caso geral)

- Seja:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- Então

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & \text{se } f(n) = O(n^{\log_b a - \epsilon}), \epsilon > 0 \\ \Theta\left(n^{\log_b a} \log n\right) & \text{se } f(n) = \Theta(n^{\log_b a}) \\ \Theta\left(f(n)\right) & \text{se } f(n) = \Omega(n^{\log_b a + \epsilon}), \epsilon > 0 \text{ e } af(n/b) \leq cf(n) \end{cases}$$

Método mestre

- O método mestre facilita a nossa vida
- Mas ele é basicamente uma mecanização dos cálculos que poderíamos fazer na mão se quisémos
- Nem sempre pode ser aplicado