

AULA 6 - ANÁLISE DA COMPLEXIDADE DE ALGORITMOS RECURSIVOS

***** Entregue, num ficheiro ZIP, este guião preenchido e o código desenvolvido *****

Implemente os seguintes **algoritmos recursivos** – sem recorrer a funções de arredondamento (floor e ceil) – e analise o **número de chamadas recursivas** executadas por cada algoritmo.

$$T_1(n) = \begin{cases} 0, & \text{se } n = 0 \\ T_1\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + n, & \text{se } n > 0 \end{cases}$$

$$T_2(n) = \begin{cases} n, & \text{se } n = 0, 1, 2, 3 \\ T_2\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + T_2\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + n, & \text{se } n > 3 \end{cases}$$

$$T_3(n) = \begin{cases} n, & \text{se } n = 0, 1, 2, 3 \\ 2 \times T_3\left(\frac{n}{4}\right) + n, & \text{se } n \text{ é múltiplo de } 4 \\ T_3\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + T_3\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + n, & \text{caso contrário} \end{cases}$$

Deve utilizar **aritmética inteira**: $n/4$ é igual a $\left\lfloor \frac{n}{4} \right\rfloor$ e $(n+3)/4$ é igual a $\left\lceil \frac{n}{4} \right\rceil$.

- **Preencha a tabela da página seguinte** com o resultado de cada função e o número de chamadas recursivas para os sucessivos valores de n .
- Analisando os dados da tabela, estabeleça uma ordem de complexidade para cada algoritmo.

$T_1(n)$ tem ordem de complexidade $O(\log(n))$

$T_2(n)$ tem ordem de complexidade $O(n)$

$T_3(n)$ tem ordem de complexidade $O(n)$

- Escreva uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função **$T_1(n)$** . Obtenha, depois, uma **expressão exata e simplificada**; determine a sua **ordem de complexidade**. Compare a expressão obtida com os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico**.

$$C(0) = 1$$

$$C(n) = C\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + k$$

$$N=0, k = 1 + \log_4 n$$

$$C(n) = 1 + C\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + 1 + \log_4 n = C(0) + 1 + \log_4 n$$

É um algoritmo de complexidade $O(\log(n))$

n	$T_1(n)$	Nº de Chamadas Recursivas	$T_2(n)$	Nº de Chamadas Recursivas	$T_3(n)$	Nº de Chamadas Recursivas
0	0	1	0	1	0	1
1	1	2	1	1	1	1
2	2	2	2	1	2	1
3	3	2	3	1	3	1
4	5	3	6	3	6	2
5	6	3	8	3	8	3
6	7	3	9	3	9	3
7	8	3	10	3	10	3
8	10	3	12	3	12	2
9	11	3	14	3	14	3
10	12	3	15	3	15	3
11	13	3	16	3	16	3
12	15	3	18	3	18	2
13	16	3	22	5	22	4
14	17	3	23	5	23	4
15	18	3	24	5	24	4
16	21	4	28	7	28	3
17	22	4	31	7	31	6
18	23	4	32	7	32	6
19	24	4	33	7	33	6
20	26	4	36	7	36	4
21	27	4	38	7	38	7
22	28	4	39	7	39	7
23	29	4	40	7	40	7
24	31	4	42	7	42	4
25	32	4	44	7	44	7
26	33	4	45	7	45	7
27	34	4	46	7	46	7
28	36	4	48	7	48	4

- Escreva uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função **$T_2(n)$** . Considere o caso particular **$n = 4^k$** e obtenha uma **expressão exata e simplificada**; determine a **ordem de complexidade** para esse caso particular. Compare a expressão obtida com os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico** e confirme o resultado obtido usando o **Teorema Mestre**.

$$\begin{aligned}
 C(0) &= C(1) = C(2) = C(3) = 1 \\
 C(n) &= 1 + C(\text{floor}(\frac{n}{4})) + C(\text{ceil}(\frac{n}{4})) + 2 \\
 n = 4^k &\rightarrow k = \log_4 n \\
 C(n) &= 3 \times C(\frac{n}{4}) + 3 = 3 \times (3 \times C(\frac{n}{16}) + 3) + 3 = 3^{k+1} - 3 \\
 C(n) &= 3^{\log_4 n + 1} - 3 = 3 \times n^{\log_4 3} - 3 \Rightarrow O(n^{\log_4 3}) \\
 \text{Teorema do Mestre} \\
 a = 3; b = 4; d = 0 &\rightarrow a > b^d \rightarrow O(n^{\log_4 3})
 \end{aligned}$$

- Pode **generalizar a ordem de complexidade** que acabou de obter para todo o n ? **Justifique.**

Como o desenvolvimento telescópico é do tipo $n^\alpha (\alpha > 0)$ a ordem de complexidade pode ser generalizada para todo o n .

- Obtenha uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função **$T_3(n)$** .

$$\begin{aligned} C(0) &= C(1) = C(2) = C(3) = 1 \\ C(n) &= 2 \times C\left(\frac{n}{4}\right) + n \rightarrow n \% 3 == 0 \\ C(n) &= C(\text{floor}(\frac{n}{4})) + C(\text{ceil}(\frac{n}{4})) + n \end{aligned}$$

- Considere o caso particular $n = 4^k$ e obtenha uma **expressão exata e simplificada**; determine a **ordem de complexidade** para esse caso particular. Compare a expressão obtida com os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico** e confirme o resultado obtido usando o **Teorema Mestre**.

$$\begin{aligned} C(0) &= C(1) = C(2) = C(3) = 1 \\ \text{Para } n \text{ múltiplo de } 3: \\ C(n) &= 2 + C\left(\frac{n}{4}\right) = 3 + C\left(\frac{n}{16}\right) = k + 1 + C\left(\frac{n}{4^k}\right) \\ \text{Substitui } k \text{ por } \log_4 n \\ C(n) &= \log_4 n + 1 + C\left(\frac{n}{4^{\log_4 n}}\right) = \log_4 n + C(1) = \log_4 n + 1 \rightarrow O(\log_4 n) \\ \text{Teorema Mestre:} \\ a &= 1; b = 4; d = 0. a = b^d \rightarrow O(n^d \log_b n) \rightarrow O(\log_4 n) \end{aligned}$$

- Pode **generalizar a ordem de complexidade** que acabou de obter para todo o n ? **Justifique.**

Como o desenvolvimento telescópico é do tipo $n^\alpha (\alpha > 0)$ a ordem de complexidade pode ser generalizada para todo o n .

- Atendendo às **semelhanças entre $T_2(n)$ e $T_3(n)$** estabeleça uma **ordem de complexidade para $T_3(n)$** . **Justifique.**

Como $T_3(n)$ e $T_2(n)$ calculam o mesmo resultado, mas $T_3(n)$ faz menos chamadas recursivas, então $T_3(n)$ não pode ter ordem de complexidade superior à de $T_2(n)$

