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.

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

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

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

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

- **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.

O algoritmo $T_1(n)$ tem ordem de complexidade logarítmica, ou seja, $O(\log n)$. O algoritmo $T_2(n)$ tem ordem de complexidade linear, $O(n)$. No algoritmo $T_3(n)$ os resultados são iguais aos do algoritmo $T_2(n)$, no entanto, o número de chamadas recursivas é mais reduzido. Logo, $T_3(n)$ terá ordem de complexidade linear, mas nunca superior à ordem de complexidade linear do algoritmo $T_2(n)$.

- Escreva uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função $\mathbf{T_1(n)}$. 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**.

Desenvolvimento telescópico, admitindo que n = 4^k:

Número de chamadas recursivas: $C_1(n) = 1 + C_1\left(floor\left(\frac{n}{4}\right)\right) = 2 + C_2\left(floor\left(\frac{n}{4^2}\right)\right) =$

$= 1 + 1 + 1 + \cdots + C_1\left(floor\left(\frac{n}{4^k}\right)\right) = k + C_1\left(floor\left(\frac{n}{4^k}\right)\right)$

Para um valor de k de $\frac{n}{4^k}$, então: $\frac{n}{4^k} = 1 \Leftrightarrow n = 4^k \Leftrightarrow k = \log_4 n$

Sendo assim, $C_1(n) = 1 + \log_4 n$

Ordem de Complexidade: $\theta(n)$

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

- Escreva uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função $\mathbf{T_2(n)}$**. Considere o caso particular** $\boldsymbol{n = 4^k}$ **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**.

Expressão recorrente: $C_2(n) = \begin{cases} 1, se\ n = 0, 1, 2, 3 \\ C_2\left(floor\left(\frac{n}{4}\right)\right) + C_2\left(ceil\left(\frac{n}{4}\right)\right) + 2, se\ n > 3 \end{cases}$

Vamos admitir que n = 4^K e que $floor\left(\frac{n}{4}\right) = ceil\left(\frac{n}{4}\right) = \frac{n}{4}$. Usando o desenvolvimento telescópico:

$C_2(n) = 2 + C_2\left(floor\left(\frac{n}{4}\right)\right) + C_2\left(ceil\left(\frac{n}{4}\right)\right) = 2 + C_2\left(\frac{n}{4}\right) + C_2\left(\frac{n}{4}\right) = 2 * C_2\left(\frac{n}{4}\right) + 2 =$

$= 2 * \left(2 * C_2\left(\frac{n}{4}\right) + 2\right) + 2 = 2 * \left[2 * \left(2 * C_2\left(\frac{n}{4}\right) + 2\right) + 2\right] + 2 =$

$= 2^K * C_2\left(\frac{n}{4^K}\right) + \sum_{i=1}^{K} 2^i = 2^K * C_2\left(\frac{n}{4^K}\right) + 2^{i-1} - 2$, o que revela que tem uma complexidade linear, uma vez que $\log_4 n = k$.

Considerando então a expressão $C_2(n) = 2 * C_2\left(\frac{n}{4}\right) + 2$, podemos saber os seguintes valores:

- ➔ $a = 2$;
- ➔ $b = 4$;
- ➔ Como $f(n)$ é uma constante, podemos então concluir que $d = 0$.

Aplicando assim o Teorema Mestre:

$a > b^d \Leftrightarrow 2 > 4^0 \Leftrightarrow 2 > 1$, logo podemos concluir que $C_2(n) = \theta\left(n^{\log_4 2}\right) = \theta\left(n^{\frac{1}{2}}\right)$, por isso tem complexidade linear, $O(n)$, tal como também foi comprovado pelos valores da tabela e pelo desenvolvimento telescópico.

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

Como o desenvolvimento telescópico da função é do tipo $n^\alpha, com\ \alpha > 0$, podemos concluir que é uma função denominada de suave. Por essa razão, a ordem de complexidade pode ser generalizada para todo o valor de n.

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

Expressão recorrente: $C_3(n) = \begin{cases} 1, & se\ n = 0, 1, 2, 3 \\ C_2\left(floor\left(\frac{n}{4}\right)\right) + 1, & se\ n\ múltiplo\ de\ 4 \\ C_2\left(floor\left(\frac{n}{4}\right)\right) + C_2\left(ceil\left(\frac{n}{4}\right)\right) + 2, & se\ caso\ contrário \end{cases}$

- **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**.

Vamos fazer o cálculo do número de chamadas recursivas tal que $n = 4^K$, para valores múltiplos de 4.
Partindo do 2º ramo da expressão recorrente, podemos utilizar o desenvolvimento telescópico, admitindo que $floor\left(\frac{n}{4}\right) = \frac{n}{4}$:
$C_3(n) = 1 + C_3\left(\frac{n}{4}\right) = 2 + C_3\left(\frac{n}{4^2}\right) = 3 + C_3\left(\frac{n}{4^3}\right) = K + C_3(\frac{n}{4^K})$, com $K = \log_4 n$, logo conclui-se que tem complexidade logarítmica.

Aplicando agora o Teorema Mestre à fórmula $C_3(n) = 1 + C_3(\frac{n}{4})$, podemos obter os seguintes valores:

- ➔ $a = 1$;
- ➔ $b = 4$;
- ➔ $f(n) = 1$, que é uma constante, logo $d = 0$.

Fazendo a igualdade $a = b^d \Leftrightarrow 1 = 4^0$, podemos afirmar que $C_3(n) = \theta(n^0 \log n) = \theta(\log n)$.

Uma vez que tanto no desenvolvimento telescópico como no Teorema Mestre a expressão de $C_3(n)$, tem complexidade $\theta(\log n)$, podemos concluir que tem complexidade logarítmica.

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

Não. Contrariamente ao que acontece nas outras funções, esta função revela um número de chamadas recursivas decrescente em alguns valores de n, nomeadamente quando esses valores são múltiplos de 4, onde se nota bastante essas oscilações de chamadas à função recursiva. Por essa razão, não podemos usar a regra da "suavidade". Por outro lado, os valores que se obteve na alínea anterior é válido para valores $n = 4^K$, logo não se pode generalizar a ordem de complexidade para todo o n.

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

Observando os valores de $T_2(n)$ e de $T_3(n)$ representados na tabela acima, podemos constatar que são bastante semelhantes. Os resultados para os valores de n são iguais, apenas diferenciando no número de chamadas recursivas. Podemos deste modo concluir que $T_2(n)$ exige um maior esforço computacional do que $T_3(n)$, uma vez que a primeira função faz mais chamadas do que a segunda função referida.
No entanto, as duas funções apresentam semelhanças e têm a mesma ordem de complexidade. Logo, não está errado afirmar que a ordem de complexidade de $T_3(n)$ é $\theta(n)$.