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.

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

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

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

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.

T1(n) tem ordem de complexidade O($\log_4 n$) → Logarítmica
T2(n) tem ordem de complexidade O($n^{\frac{1}{2}}$) → Aproximação linear
T3(n) tem ordem de complexidade O($n^{\frac{1}{2}}$) → Aproximação linear

- 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:

Sendo C(n) o número de chamadas recursivas de T1(n):

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

C(n)=C(n/4) + 1
C(n)=C(n/4^2) + 1 + 1
C(n)=C(n/4^3) + 1 + 1 + 1
C(n)=C(n/4^4) + 1 + 1 + 1 + 1
C(n)=C(n/4^k) + k

Como n/4^k = 1 → k = $\log_4 n$ logo,

C(n) = 1+ $\log_4 n$ → 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**.

- Desenvolvimento telescópico:

Sendo C(n) o número de chamadas recursivas de T2(n):

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

C(n)= 2^1C(n/4) + 2
C(n)= 2^2C(n/4^2) + 4 + 2
C(n)= 2^3C(n/4^3) + 8 + 4 + 2
C(n)= 2^kC(n/4^k) + $2^{k+1} - 2$

Como n/4^k = 1 → k = $\log_4 n$ logo,

C(n) = $2^k + 2^{k+1} - 2 = 2^k(1 + 2) - 2 = 2^k * 3 - 2 = 3 * 2^{\log_4(n)} - 2 = 3\sqrt{n} - 2$ = O($n^{\frac{1}{2}}$)

- Confirmação do resultado através do Teorema Mestre:

C(n) = 2C(n/4) +2 = a(n/b) + f(n)

a= 2
b= 4
f(n) = 2 ou seja f(n) = constante, logo d =0.

Como 2 >4^0 → a>b^d então O($n^{\log_b a}$) = O($n^{\log_4 2}$) = O($n^{\frac{1}{2}}$)

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

Neste caso posso generalizar a ordem de complexidade que obtive para todo o "n" visto que, ao contrário da função T3(n), nesta função temos apenas um ramo que conta o número de chamadas recursivas e por isso não há valores de "n" em que sejam feitas mais ou menos chamadas recursivas logo o número de chamadas recursivas depende apenas desse ramo.

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

Sendo C(n) o número de chamadas recursivas de T3(n):

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

- **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**.

- Desenvolvimento telescópico:

Sendo C(n) o número de chamadas recursivas de T3(n):

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

C(n)= C(n/4) + 1
C(n)= C(n/4^2) + 1 + 1
C(n)= C(n/4^3) + 1 + 1 + 1
C(n)= C(n/4^k) + k

Como n/4^k = 1 → k = $\log_4 n$ logo,

C(n) = 1 + $\log_4 n$ = 1 + $\log_4 n$ = O($\log n$)

- Confirmação do resultado através do Teorema Mestre:

C(n)= C(n/4) + 1 = a(n/b) + f(n)

a= 1
b= 4
f(n)=1 ou seja f(n) = constante, logo d =0.

Como 1=4^0 → a = b^d então O($n^d \log n$) = O($n^0 \log n$) = O($\log n$)

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

Neste caso não posso generalizar a ordem de complexidade que obtive para todo o "n" uma vez que a ordem de complexidade obtida se aplica apenas quando o "n" é múltiplo de 4 e este caso ocorre menos vezes do que o caso em que o "n" não é múltiplo de 4. Outra razão pela qual não posso generalizar é o facto de que se o fizesse estaria a considerar apenas os casos do algoritmo em que são feitas menos chamadas recursivas, ou seja, estaria a considerar os melhores casos.

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

De acordo com a questões anteriores a ordem de complexidade da função T3(n) quando o n é múltiplo de 4 é igual à ordem de complexidade da função T1(n) (= O($\log n$)) .
No entanto, como a função T3(n) se comporta da mesma forma relativamente ao número de chamadas recursivas que a função T2(n) quando o n não é múltiplo de 4 e este caso ocorre a maior parte das vezes conclui-se que a ordem de complexidade da T3(n) é $\mathbf{O(n^{\frac{1}{2}})}$.

Conclui-se, então, que apesar das funções T2(n) e T3(n) terem a mesma ordem de complexidade a função T3(n) consegue ser mais eficiente visto que poupa operações sempre que o "n" é múltiplo de 4.