NOME:	ANDRÉ PATACAS	N.º MEC:	93357

AULA 5 - 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.

$$T_1(n) = \begin{cases} 0, \text{se } n = 0 \\ T_1\left(\left\lfloor \frac{n}{3} \right\rfloor\right) + n, \text{se } n > 0 \end{cases}$$

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

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

Deve utilizar **aritmética inteira**: n/3 é igual a $\left\lfloor \frac{n}{3} \right\rfloor$ e (n+2)/3 é igual a $\left\lceil \frac{n}{3} \right\rceil$.

- **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 T1 tem ordem de complexidade logarítmica O(log(n)).
O algoritmo T2 tem ordem de complexidade linear O(n)
O algoritmo T3 por pertencer a uma classe de complexidade igual ou inferior a T2 também pertence a O(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 a os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico**.

C(0) = 0
C(n) = 1 + C(floor(n/3)) = 2 + C(floor(n/9)) = … = k + C(floor(n/3^k)),

para k = floor(log3(n))
C(n) = floor(log3(n)) + C(1) = floor(log3(n)) + 1

Resultado igual ao da tabela

n	T1(n)	Nº de Chamadas Recursivas	T2(n)	Nº de Chamadas Recursivas	T3(n)	Nº de Chamadas Recursivas
0	0	0	0	0	0	0
1	1	1	1	0	1	0
2	2	1	2	0	2	0
3	4	2	5	2	5	1
4	5	2	7	2	7	2
5	6	2	8	2	8	2
6	8	2	10	2	10	1
7	9	2	14	4	14	3
8	10	2	15	4	15	3
9	13	3	19	6	19	2
10	14	3	22	6	22	5
11	15	3	23	6	23	5
12	17	3	26	6	26	3
13	18	3	28	6	28	6
14	19	3	29	6	29	6
15	21	3	31	6	31	3
16	22	3	34	6	34	5
17	23	3	35	6	35	5
18	26	3	38	6	38	2
19	27	3	43	8	43	6
20	28	3	44	8	44	6
21	30	3	49	10	49	4
22	31	3	51	10	51	8
23	32	3	52	10	52	8
24	34	3	54	10	54	4
25	35	3	59	12	59	7
26	36	3	60	12	60	7
27	40	4	65	14	65	3
28	41	4	69	14	69	9

- Escreva uma **expressão recorrente** para o **número de chamadas recursivas** efetuadas pela função **$T_2(n)$. Considere o caso particular $n = 3^k$** e obtenha uma **expressão exata e simplificada;** determine a **ordem de complexidade** para esse caso particular. Compare a expressão obtida com a os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico** e confirme o resultado obtido usando o **Teorema Mestre**.

C(0) = C(1) = C(2) = 0
Para n = 3^k
C(n) = 2 + 2 * C(n/3) = 2 + 2 * (2 + 2 * C(n/9)) = 2 + 4 + 4 * C(n/9) = 2 + 4 + 8 + 8 * C(n/27) =
… = 2^(k + 1) – 2 + 8 * C(n/3^k)

Para k = log_3(n)

C(n) =2 * (2 ^ log_3(n) – 1) + C(1) = 2 * (2 ^ log_3(n) – 1)
Como a^(log_b(c)) == c^(log_b(a))
C(n) = 2 * (n ^ log_3(2) – 1)

C(n) = 2 + 2 * C(n/3), pelo teorema mestre e pela regra da suavidade conclui-se que T2 ∈ O(n^log_3(2))
log_3(2) ~= 0.95
O que concorda com os dados da tabela e com a previsão feita

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

O comportamento da função é "regular" o que permite pela regra da suavidade generalizar a conclusão para todo o n

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

C(n) = 0	, n == 0, 1, 2
1 + C(n/3)	, n % 3 == 0
2 + C(floor(n/3)) + C(ceil(n/3))	, (n != 0, 1, 2 || n % 3 != 0)

- **Considere o caso particular $n = 3^k$** e obtenha uma **expressão exata e simplificada;** determine a **ordem de complexidade** para esse caso particular. Compare a expressão obtida com a os dados da **tabela**. Sugestão: use o **desenvolvimento telescópico** e confirme o resultado obtido usando o **Teorema Mestre**.

Para n = 3^k, n % 3 == 0 =>

C(n) = 0	, n == 0, 1, 2
1 + C(n/3)	, n % 3 == 0

C(n) = 1 + C(n/3) = 2 + C(n/9) = … = log_3(3^k) + C(n/3^k) = log_3(n) + C(1) = log_3(n)
⇨ T3 pertence a O(log(n))

Pelo teorema mestre
C(n) = a C(n / b) + f(n) =>
a = 1, b = 3, c = 0, d e O(n^d) => d = 0, b^d = 1 =>
a == b^d =>

T3 pertence à classe de complexidade O(n^d log(n)) == O(n^0 log(n)) == O(log(n))
O desenvolvimento telescópico e o teorema mestre, como esperado, resultaram na mesma classe de complexidade

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

A ordem de complexidade foi obtida para um caso especifico da função T3 e o seu comportamento em casos que não n = 3^k é muito distinto não sendo por isso possível generalizar o resultado obtido.

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

Na melhor das hipóteses T2 tem ordem de complexidade igual à de T3, ou seja, utilizando um abuso de linguagem, O(T3) <= O(T2), o que permite concluir que a ordem de complexidade de T3 é, no pior dos casos, igual à de T2, O(n^log_3(2))